НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмна система для визначення комплексної оцінки відгуків Інтернет-користувачів»

Виконав:
студент IV курсу, групи КП-62
Лук'янець Михайло Олександрович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Тетяна Миколаївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ММСА ІПСА, к.т.н., доцент,
Дідковська Марина Віталіївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

«ЗАТВЕРДЖУЮ»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Лук'янцю Михайлу Олександровичу

1. Тема проєкту «Програмна система для визначення комплексної оцінки відгуків Інтернет-користувачів», керівник проекту Заболотня Тетяна Миколаївна, к.т.н., доцент, затверджені наказом по університету від «25» травня 2020 р. № 1181-с

2. Термін подання студентом проєкту «12» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих рішень та постановка задачі;
   - обґрунтування вибору засобів реалізації;
   - структурно-алгоритмічна організація системи;
   - особливості реалізації програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   - основний алгоритм роботи системи (креслення);
   - алгоритм аналізу текстових відгуків (креслення);
   - структура ситеми (плакат);

– дерево проблем (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	14.11.2019	
2.	Розроблення та узгодження технічного завдання	28.11.2019	
3.	Розроблення структури програмної системи	15.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2019	
5.	Розроблення архітектури системи та інтерфейсу	03.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	20.02.2020	
7.	Програмна реалізація програмної системи	10.03.2020	
8.	Тестування програмної системи	17.03.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	11.04.2020	
11.	Підготовка графічної частини дипломного проєкту	21.04.2020	
12.	Оформлення документації дипломного проєкту	26.05.2020	

Студент Михайло ЛУК’ЯНЕЦЬ

Керівник проєкту Тетяна ЗАБОЛОТНЯ

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробленню програмної системи для аналізу відгуків Інтернет-користувачів.

Розроблена програмна система являє собою систему збору відгуків з мережі Інтернет з декількох джерел, їх обробки та аналізу й інтерфейсу користувача, який являє собою телеграм бота. Даний інтерфейс є єдиним розробленим інтерфейсом, проте архітектура системи створена з огляду на можливість для створення інших інтерфейсів. Функціональність системи забезпечує доступ користувачів месенджеру Telegram у вигляді чат-боту з певним набором команд для маніпулювання набором посилань на ресурси, з яких збираються та аналізуються дані. Аналіз емоційного забарвлення відгуків здійснюється за допомогою нейронної мережі за системою емоцій PERMA. Нейронна мережа базується на бібліотеці Keras та моделі bert, що використовується для ембедингів.

У даному дипломному проєкті розроблено: архітектуру системи аналізу відгуків, модулі збору відгуків інтернет-користувачів, алгоритм передобробки текстів для аналізу нейронною мережею, нейронну мережу для оцінки тексту за набором з п’яти емоцій та інтерфейс користувача у вигляді телеграм боту.

ABSTRACT

This diploma project is devoted to the development of a software system for analyzing the feedback of Internet users.

The developed software system is a system for collecting feedback from the Internet from several sources, their processing, analysis and the user interface, which is a telegram bot. This interface is the only interface developed, but the system architecture is designed to create other interfaces. The functionality of the system provides users with access to the Telegram messenger in the form of a chatbot with a specific set of commands to manipulate a set of links to resources from which data is collected and analyzed. The analysis of emotional coloring of responses is carried out by means of a neural network on system of emotions PERMA. The neural network is based on the Keras library and the bert model used for embeddings.

This thesis project has developed: the architecture of the feedback analysis system, modules for collecting feedback from Internet users, an algorithm for text processing for neural network analysis, a neural network for evaluating text on a set of five emotions and a user interface in the form of a bot telegram.

ДП.045440-01-90 Програмна система для визначення комплексної оцінки відгуків Інтернет-користувачів. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Програмна система для	5	
	визначення комплексної		
	оцінки відгуків		
	Інтернет-користувачів.		
	Технічне завдання		
ДП.045440-03-81	Програмна система для	54	
	визначення комплексної		
	оцінки відгуків		
	Інтернет-користувачів.		
	Пояснювальна записка		
ДП.045440-04-51	Програмна система для	4	
	визначення комплексної		
	оцінки відгуків		
	Інтернет-користувачів.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмна система для	9	
	визначення комплексної		
	оцінки відгуків		
	Інтернет-користувачів.		
	Керівництво користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Програмна система для	1	
	визначення комплексної		
	оцінки відгуків		
	Інтернет-користувачів.		
	Узагальнений алгоритм		
	роботи системи.		
	Схема алгоритму		
ДП.045440-07-99	Програмна система для	1	
	визначення комплексної		
	оцінки відгуків		
	Інтернет-користувачів.		
	Аналіз текстових відгуків.		
	Схема алгоритму		
ДП.045440-08-98	Програмна система для	1	
	визначення комплексної		
	оцінки відгуків		
	Інтернет-користувачів.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2019 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ КОМПЛЕКСНОЇ ОЦІНКИ ВІДГУКІВ ІНТЕРНЕТ-КОРИСТУВАЧІВ

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Михайло ЛУК'ЯНЕЦЬ

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмна система для визначення комплексної оцінки відгуків Інтернет-користувачів.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

Розробка виконана на замовлення Центру електронної освіти КПІ ім. Ігоря Сікорського (Договір №24-8 від 15.09.2010).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості допоміжного інструменту аналізу відгуків Інтернет-користувачів з метою отримання багатофакторної оцінки опіній користувачів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмна система повинна забезпечувати такі основні функції:

1) можливість редагування списку посилань для пошуку відгуків;
2) збір відгуків;
3) аналіз цифрової оцінки відгуків;
4) аналіз емоційного забарвлення відгуків за системою PERMA;
5) формування звіту про аналіз відгуків;
6) формування звіту на основі історії збору відгуків.

Розробку виконати на платформі Python з використанням технології NLTK та Keras.

Додаткові вимоги:

1) інтерфейс у вигляді телеграм боту;

2) наявність інструкції з командами у інтерфейсі;

3) наявність логотипу для бота;

4) звіти з графіками, що мають легенду та сітку.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

- «Алгоритм аналізу текстових відгуків. Схема алгоритму»;
- «Основний алгоритм роботи системи. Схема роботи програмної системи».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи……..14.11.2019

Розроблення та узгодження технічного завдання……....28.11.2019

Розроблення структури програмної системи…….....15.12.2019

Розроблення архітектури системи та інтерфейсу……03.02.2020

Програмна реалізація програмної системи……...……....17.03.2020

Тестування програмної системи……...…….....03.04.2020

Підготовка матеріалів текстової частини проєкту……..….28.04.2020

Підготовка матеріалів графічної частини проєкту……...…12.05.2020

Оформлення технічної документації проєкту……..…25.05.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2020 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ КОМПЛЕКСНОЇ ОЦІНКИ ВІДГУКІВ ІНТЕРНЕТ-КОРИСТУВАЧІВ

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Михайло ЛУК'ЯНЕЦЬ

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

NLP – Natural Language Processing – загальний напрям інформатики, штучного інтелекту та математичної лінгвістики. Він вивчає проблеми комп'ютерного аналізу та синтезу природної мови.

LINQ – Language Integrated Query – компонент Microsoft .NET Framework, який додає нативні можливості виконання запитів даних до мов, що входять у .NET.

API – Application Programming Interface (укр. «інтерфейс прикладного програмування») – набір стандартів та методів взаємодії окремих компонентів програмної системи.

CLR – Common Language Runtime – це компонент стандартного пакету Microsoft .NET Framework, віртуальна машина, на якій виконуються всі мови платформи .NET Framework.

Фреймворк – набір абстракцій, які надають базові можливості програмного забезпечення для подальшої композиції та модифікації.

HTTP – Hypertext Transfer Protocol (укр. «протокол передачі гіпер- текстових документів»), протокол прикладного рівня для передачі даних у комп’ютерних мережах.

HTML – мова розмітки гіпертекстових документів, мова розмітки для створення веб-сторінок.

CSS – каскадні таблиці стилів, описують яким чином елементи HTML розташовуються та виглядають на екрані.

CUDA– Compute Unified Device Architecture – програмно-апаратна архітектура паралельних обчислень, яка дозволяє істотно збільшити обчислювальну продуктивність завдяки використанню графічних процесорів фірми Nvidia.

GPU – Graphics Proccesing Unit.

JSON – текстовий формат обміну даними.

БД – база даних.

IDE – Integrated Development Environment.

JS – Java Script.

ООП – об’єктно-орієнтовне програмування.

TPL – Task Parallel Library.

XML – Extensible Markup Language.

MSIL – Microsoft Intermediate Language.

LLVM – Low Level Virtual Machine.

ВСТУП

XXI сторіччя стало часом надзвичайного поширення мережі Інтернет. Кількість користувачів більша, ніж декілька мільярдів. Кількість пристроїв для доступу до мережі Інтернет у кожної людини в розвинених країнах часто більша, або й набагато більше, ніж один. Проникнення Інтернету в повсякденне життя лише збільшується з часом. Наприклад, продажі ігор для платформ ПК та консолей вже перебільшують 80%. Ігри для телефонів можливо придбати виключно онлайн. Закономірним є й збільшення продажів усіх інших товарів через онлайн-магазини.

У зв’язку з цим, доступ користувача до можливих продуктів вжитку надзвичайно збільшився. Тисячі та десятки тисяч фільмів, ігор, музики, різноманітних пристроїв та інших товарів доступні для вибору у зручному форматі та не вимагають майже жодних зусиль для їх пошуку. Разом з цим постала інша проблема – проблема вибору товарів серед надзвичайної кількості варіантів для вибору.

Для вирішення вже цієї проблеми з’явилися сайти, що дозволяють залишити свій відгук на певний товар або його продавця чи виробника. Прикладами ресурсів, на яких користувачі залишають відгуки, є IMDB, Amazon, Metacritic, Google Maps, Rozetka та багато інших. На основі таких відгуків компанії з продажу товарів чи послуг формують рейтинги продуктів. В свою чергу, дані про рейтинг можуть суттєво впливати на вибір покупця. При цьому слід відмітити, що цікавою і для компаній, і для покупців є наявність відгуків про один і той самий продукт на різних ресурсах.

Отже, є актуальною задача автоматизованого збирання та аналізу відгуків користувачів одразу з декількох джерел. Така система може вирішити завдання аналізу наявних відгуків у мережі Інтернет. Одним з варіантів вирішення проблеми аналізу відгуків з різних ресурсів, що виводять оцінки за алгоритмами, що можуть відрізнятися для кожного

ресурсу, може бути система аналізу емоційного забарвлення тексту відгуків. Такий аналіз відгуків робить можливим визначення того, чи є відгук схвальним чи навпаки, несхвальним, та, користуючись цими даними, сформувати певну оцінку товару.

Даний дипломний проєкт присвячений розробленню програмної системи для визначення комплексної оцінки відгуків Інтернет-користувачів.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз проблеми оцінки відгуків Інтернет користувачів

Відгук Інтернет-користувача є відгуком, написаним користувачем або споживачем продукту чи послуги на основі досвіду як користувача рецензованого продукту.

Популярними джерелами відгуків споживачів є сайти електронної комерції, такі як Amazon.com, сайти-довідники про фільми та серіали, наприклад IMDB.com, та сайти соціальних медіа, як TripAdvisor та Yelp.

На сайтах електронної комерції часто є відгуки споживачів щодо товарів та продавців окремо. Зазвичай відгуки споживачів мають форму декількох рядків текстів, що супроводжуються числовою оцінкою.

Цей текст покликаний допомогти у прийнятті рішення про покупки потенційного покупця. Споживчий огляд товару зазвичай коментує, наскільки добре продукт відповідає очікуванням на основі специфікацій, наданих виробником чи продавцем. Це говорить про продуктивність, надійність, дефекти якості, якщо такі є, та співвідношення ціни та якості.

Відгук користувача, який також називається «контент, створений користувачем», відрізняється від «контенту, створеного маркетингом», в його оцінці з точки зору споживача чи користувача. Часто вона включає порівняльні оцінки щодо конкуруючих продуктів.

Спостереження мають як фактичний, так і суб'єктивний характер. Відгук споживачів про продавців зазвичай коментує досвід роботи та надійність або надійність продавця. Зазвичай в ньому коментуються такі фактори, як своєчасність доставки, упаковки та правильність доставлених товарів, транспортні збори, послуги повернення проти обіцяних обіцянок тощо.

Огляди споживачів в Інтернеті стали головним фактором ділової репутації та іміджу бренду завдяки популярності веб-сайтів TripAdvisor, Yelp та відгуків на інших веб-сайтах.

Негативний огляд може завдати шкоди репутації бізнесу, і це створило нову галузь управління репутацією, де компанії намагаються видалити або приховати погані відгуки, щоб знайти більш сприятливий контент, коли потенційні клієнти проводять дослідження.

Споживачі сприймають відгуки користувачів, використовуючи правильну граматику та переконливий стиль написання, кращої якості, ніж відгуки, написані іншими способами.

Зв'язок між відгуками користувачів та якістю товару однозначно невизначений. Іноді за певних обставин може не бути зв’язку між якістю товару чи послуги та їх рейтингами.

Для товарів найвищого рівня якості, одне дослідження виявило, що рейтинги користувачів збігаються з професійними рейтингами трохи більше половини часу. Крім того, люди, які читають відгуки користувачів, як правило, сприймають їх так само об'єктивно, як і наукове тестування, особливо коли є середній бал відгуків користувачів.

Використовуючи спеціальні алгоритми аналізу текстів для великої кількості відгуків можливо досить точно визначити які відгуки створені одними і тими самими авторами.

Відповідно, через велику кількість відгуків, що зростає кожного року, та необхідність моніторингу за відгуками на компанію, розвиваються методи аналізу відгуків.

Такими методами є, наприклад:

- ручний моніторинг за відгуками людьми;
- статистична оцінка на основі кількості відгуків, популярності відгуків та цифрової оцінки, що виставив користувач;
- аналіз емоційного забарвлення тексту відгуку користувача за допомогою машинного навчання та ін.

Ручний моніторинг за відгуками є найпершим способом, що використовується. Не дивлячись на свої недоліки, він все ще є досить популярним, оскільки підходить для моніторингу за малою кількістю

відгуків та не вимагає використання особливого програмного забезпечення. Проте, зі зростанням кількості відгуків, поступово стає невигідним для використання через неможливість найняти достатню кількість працівників для оцінки відгуків за прийнятну ціну.

Переваги:

- відносно дешевий для малої кількості відгуків;
- достатня якість моніторингу та оцінки відгуків для малої кількості відгуків;
- можливість одночасно відповідати на відгуки користувачів.

Недоліки:

- дорогий для великої кількості відгуків;
- дуже низька ефективність оцінкювання великої кількості відгуків;
- спотворення результатів аналізу підробленими відгуками;
- потреба у великій кількості часу для проведення повторного моніторингу, оскільки він не є автоматизованим.

Статистична оцінка зазвичай сама виводиться у тому чи іншому вигляді на веб-сайтах, де можла залишити відгук із чисельною оцінкою. Завдяки цьому методу отримуються, наприклад, оцінки для товарів на веб-сайтах, що продають товари, та оцінка фільмів та серіалів на сайтах, що відповідно спеціалізуються на поданні інформації про останні.

Переваги статистичного оцінювання:

- швидке;
- автоматизований;
- виводить об’єктивну оцінку;
- здатен оброблювати велику кількість відгуків.

Недоліки:

- на малій кількості відгуків не є точним;
- підроблені відгуки спотворюють результати аналізу;

- формули для отримання оцінки можуть досить сильно відрізнятися на різних веб-сторінках, що впливає на отриману оцінку;
- аналізує лише цифрову оцінку, відповідно покладається на здатність користувачів до оцінювання у цифровому вигляді.

Аналіз емоційного забарвлення тексту стосується використання алгоритмів оброблення мови, аналізу тексту, обчислювальної лінгвістики та біометрики для систематичної ідентифікації, кількісної оцінки та вивчення афективних станів та суб'єктивної інформації.

Аналіз почуттів широко застосовується до таких матеріалів як, огляди та відповіді на опитування, дописи в Інтернет та соціальних медіа, а також матеріали від додатків, направленість яких варіюються від маркетингу до обслуговування клієнтів та клінічної медицини.

Переваги аналізу емоційного забарвлення тексту:

- можливість аналізу саме тексту з отриманням цифрової оцінки для подальшого використання у методах статистичної обробки;
- досить швидка оцінка відгуків.

Недоліки:

- потреба у збиранні даних для навчання;
- можливе спотворення підробленими відгуками результатів аналізу;
- наявність похибки при оцінюванні емоційного забарвлення;
- складність оцінки відгуків з сарказмом.(соединила б єтот пункт с предідущим)

1.2. Аналіз існуючих програмних рішень

1.2.1. SemanticForce

SemanticForce – одна з найбільш розвинутих систем збору та аналізу відгуків на сьогоднішній день [1]. Розробники позиціонують її як сервіс моніторингу і аналізу Інтернет-медіа.

Перевага сервісу полягає в тому, що збирання інформації проходить в режимі реального часу. Алгоритми SemanticForce відстежують будь-які згадування про компанію і бренд на новинних сайтах, в соціальних мережах (включаючи Instagram і YouTube), блогах і мікроблогах, на форумах та інших ресурсах, навіть якщо немає RSS.

Сервіс доступний в англомовній та російськомовній версіях, аналізує Інтернет-ЗМІ майже в 150 країнах світу.

При моніторингу враховуються прямі і непрямі згадування Під непрямими згадуваннями маються на увазі ті висловлювання, де назва бренду або продукту не називається, але з контексту стає зрозуміло, про кого або про що йде мова.

Одна з корисних функцій сервісу – аналіз цінності і впливовості повідомлення. Аналітичні звіти за підсумками моніторингу можливо робити самостійно або за певну плату віддати фахівцям SemanticForce.

Сервіс платний, але протягом 14 днів його можливо використовувати безкоштовно.

До переваг сервісу можна віднести велику кількість оброблюваних сайтів та ресурсів на декількох мовах. Також вагомою перевагою є моніторинг згадувань.

До недоліків сервісу належить відсутність аналізу емоційного забарвлення тексту та цифрових оцінок користувачів з виведенням статистики про відгуки.

1.2.2. AppFollow

Інструмент AppFollow орієнтований на компанії, що займаються розробленням і продажем мобільних додатків або використовують їх у своїй роботі [2].

Користувачі, які завантажили додаток, залишають відгуки, коментарі або претензії на сервісах AppStore, Windows Store і Google Play.

Використання AppFollow спрощує роботу з цією інформацією, тому що вся вона стікається в одне місце – аккаунт в сервісі.

Дані оброблюються в режимі реального часу, щодня або щотижня надходить звіт на електронну пошту по країнам, ключовими словами, датою, іншим параметрам.

Сервіс доступний в безоплатній та платній версіях. У безкоштовному пакеті обмежена кількість функцій, вартість платних пакетів – від 9 доларів на місяць.

Інтерфейс AppFollow доступний англійською та українськими мовами.

До переваг сервісу можна віднести високу підготовленість до відгуків до мобільних додатків та зручну роботу з цими відгуками.

До недоліків сервісу можна віднести нездатність працювати з будь-якими відгуками, окрім як до мобільних додатків, та відсутність емоційного аналізу тексту відгуку користувача.

1.2.3. Simpoll

Сервіс Simpoll спеціалізується на створенні опитувань для користувачів. Його зручність полягає в тому, що можна створювати опитування будь-якої складності за кілька хвилин: опитування-анкети, опитування-голосування, опитування-тестування [3].

Розробники сервісу беруть на себе певні технічні нюанси – від створення і публікації опитування до збору і аналізу результатів. Використовувати сервіс можливо без реєстрації – досить облікового запису в соціальній мережі.

Опитування легко вбудовується на сторінку сайту або блогу, доступне на кількох мовах, включаючи українську та англійську. Проводити дослідження можливо також через соціальні мережі, електронну розсилку, форуми. Для цього потрібно опублікувати пряме посилання на будь-якому з перерахованих ресурсів.

Simpoll пропонує кілька тарифів, вартість базового – 150 гривень на місяць.

До переваг даного сервісу відноситься аналіз отриманих цифрових даних. До недоліків відноситься робота з лише отриманими безпосередньо у певному голосуванні від даного додатку даних.

1.2.4. JagaJam

JagaJam – просунутий інструмент для відстеження відгуків і коментарів користувачів в соціальних мережах. Він охоплює всі популярні соціальні медіа, включаючи Facebook та Instagram [4].

Сервіс розроблявся для оперативної взаємодії з цільовою аудиторією, моментальної реакції на відгуки та коментарі. У JagaJam доступні інструменти для аналізу аудиторії і її активності, ступеня інтересу контенту і залученості в нього користувачів.

За допомогою інструменту легко підрахувати кількість ботів в групі або на фан-сторінці.

Також за допомогою JagaJam можна проводити конкурентний і галузевої аналізи, проводити порівняльний аналіз спільнот в соціальних мережах, виявляти пересічні аудиторії і використовувати слабкі місця конкурентів з вигодою для своєї компанії. Розробники пропонують пробний тариф – протягом 14 днів можливо користуватися інструментами сервісу безкоштовно.

До переваг сервісу можна віднести зручність у роботі з відгуками для швидкої реакції на відгуки та доступність його для популярних соціальних мереж.

До недоліків можна віднести наявність лише соціальних мереж, роботу з відгуками в основному для реакції на них, через що аналіз відгуків не має багато функцій для роботи з відгуками.

1.2.5. IQBuzz

IQBuzz – сервіс, який займається моніторингом згадувань бренду і продукції на сайтах, в Інтернет-ЗМІ і соціальних мережах [5].

Він охоплює більше 10 тисяч джерел, включаючи записи в LiveJournal і LiveInternet, коментарі та посилання в YouTube, фотографії та коментарі в Instagram.

Особливість автоматичного сервісу полягає в тому, що моніторинг відбувається в реальному часі.

Сервіс IQBuzz корисний PR-менеджерам, що відстежує ефективність кампанії по просуванню. З його допомогою можливо визначити лідерів думок, майданчики для розміщення інформації про бренд і товари з найбільшим призначеним для користувача відгуком, ефективніше аналізувати діяльність конкурентів і відслідковувати негативні відгуки, що були замовлені.

У IQBuzz доступні 4 тарифних плани, стартовий вартує 7,9 тисяч гривень на місяць. Протягом 7 днів можливо користуватися сервісом безкоштовно.

До переваг можна віднести велику кількість доступних джерел відгуків та велику кількість звітів, які можна згенерувати.

До недоліків можна віднести високу ціну, відсутність емоційного аналізу тексту відгуків та відсутність можливості додати сайт для збору відгуків.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

Для створення системи багатофакторного аналізу відгуків Інтернет-користувачів необхідно обрати мову програмування, середовище розроблення та засоби створення інтерфейсу користувача.

2.1. Вибір мови програмування

Дослідивши приблизний набір можливостей та умови для їх реалізації, було сформовано вимоги до мови програмування. Серед вимог можна виділити такі як:

- висока швидкість розроблення прототипу;
- простота розроблення прототипу;
- наявність бібліотек для створення модулю емоційного аналізу відгуків;
- активний розвиток можливостей мови для розроблення і підтримки програми у майбутньому.

2.1.1. C#

C# – мультипарадигмальна мова програмування загального призначення, що є сильно типізованою, імперативною, декларативною, функціональною, загальною, об'єктно-орієнтованою (на основі класів) та компонентно-орієнтованою [6]. Вона була розроблена близько 2000 року Microsoft в рамках своєї ініціативи .NET і пізніше затверджена як міжнародний стандарт Ecma (ECMA-334) та ISO (ISO / IEC 23270: 2018).

Mono – назва вільного проєкту з відкритим кодом з розробки компілятора та часу виконання мови. C # – одна з мов програмування, розроблена для загальної мовної інфраструктури (CLI).

C# була розроблена Андерсом Хейлсбергом, а його команду з розробки в даний час очолює Мадс Торгерсен. Найновіша версія – 8.0, яка вийшла у 2019 році разом із Visual Studio 2019 версії 16.3.

C# відноситься до сім'ї мов з C-подібним синтаксисом. З даного сімейства мов синтаксис C# найбільш близький до синтаксису C++ та Java. Мова програмування C# підтримує:

- поліморфізм;
- перевантаження операторів (включаючи явний та неявний оператор приведення типу);
- делегати;
- атрибути.;
- події;
- властивості;
- generics;
- ітератори;
- TPL;
- локальні функції;
- кортежі;
- порівняння зі зразком;
- анонімні функції з підтримкою замикань;
- LINQ;
- виняткові ситуації;
- коментарі в форматі XML.

Крім того мова використовує автоматичне керування пам'яттю.

Існує декілька реалізацій C#:

- реалізація Microsoft Visual C# – реалізація, яка найчастіше використовується. Саме вона включена до складу стандартної версії для поширення;
- реалізація Microsoft .Net Core – оновлена реалізація, що не повністю підтримує можливості .Net Framework, проте дозволила відійти від підтримки старих функцій. Завдяки цьому її вдалося

розробити для багатьох платформ. Саме вона підтримує найновіші версії мови;

- проєкт Mono – складається з таких елементів як компілятор для C# та імплементація CLI, та відповідно включає всі бібліотеки, які вимагає специфікація ECMA. Версія 6.0.0 реалізує бібліотеки класів .NET до версії мови C# 8.0.

Переваги:

- мова активно розвивається та має план розвитку на деякий час вперед;
- MSDN – велика база з документацією по кожному класу та методу з прикладами їх використання та посиланнями на інші можливі джерела інформації по даному методу чи класу;
- автоматичне керування пам’яттю;
- велика стандартна бібліотека;
- LINQ – ітератори, вбудовані у мову;
- досить висока швидкість роботи;
- велика кількість інструментів для розробки багатопотокових програм, найновіший серед яких таск та async/await;
- можливість програмувати у функціональному стилі, нові функціїї з F#(функціональна мова програмування, що належить до .NET та цим є близькою до C#) постійно додаються до мови;
- Roslyn – компілятор, повністю розроблений на мові C#. Являє собою «компілятор як сервіс», що дозволяє набагато більше ніж просто компілювати код. З’явився у C# починаючи з шостої версії;
- гарна кросплатформенна підтримка;
- наявність бібліотек для розробки застосунків з машинного та глибокого навчання.

Недоліки:

- закрита реалізація;
- кількість бібліотек для машинного та глибокого навчання не дуже активно зростає, а наявні розвиваються не з дуже великою швидкістю;
- вимагає досить багато коду для реалізації;
- досить складна для входу, оскільки за багато років існування активно розвивалася та стала підтримувати багато різних бібліотек та можливостей, через що є надзвичайно велика кількість інструментів для однієї і тієї ж самої задачі;
- найбільша вбудована бібліотека міститься лише в .NET версії.

2.1.2. Java Script

JavaScript – мова програмування, яка відповідає специфікації ECMAScript. JavaScript – високорівнева, часто компільована під час виконання та багатопарадигмальна. Вона має синтаксис з фігурними дужками, динамічне введення тексту, орієнтовану на прототип об'єктну орієнтацію та функції першого класу [7].

Поряд з HTML та CSS, JavaScript є однією з основних технологій всесвітньої павутини. JavaScript включає інтерактивні веб-сторінки і є невід'ємною частиною веб-додатків. Переважна більшість веб-сайтів використовують її на стороні клієнта, а всі основні веб-браузери мають спеціальний механізм для виконання скриптів на мові JavaScript.

Як мова для багатьох парадигм, JavaScript підтримує керовані подіями, функціональні та імперативні стилі програмування. Вона має інтерфейси прикладного програмування (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних та Моделлю об'єкта документа (DOM). Однак сама мова не включає жодного вводу / виводу, такого як мережеві засоби, сховища чи графічні засоби, оскільки хост-середовище (як правило, веб-браузер) надає ці API.

Особливості мови JavaScript:

- імперативна та структурована;
- слабо типізована;
- динамічна;
- об’єктно орієнтована;
- функціональна;
- регулярні вирази;
- promise;
- особливості, що залежать від постачальника, такі як умовний catch та інші.

Рушії JavaScript спочатку використовувались лише у веб- браузерах, але тепер вони вбудовані в деякі сервери, як правило, через Node.js. Вони також вбудовані в різноманітні програми, створені за допомогою таких систем, як Electron та Cordova.

Хоча між JavaScript та Java є схожість, включаючи назву мови, синтаксис та відповідні стандартні бібліотеки, обидві мови відрізняються та сильно відрізняються за дизайном.

Переваги:

- кросплатформна;
- може використовуватися як для фронтенду, так і для бекенду;
- підходить для створення прототипів;
- має велику кількість фреймворків;
- відносно проста;
- має велику вбудовану стандартну бібліотеку функцій;
- нові версії з’являються досить часто;
- робота з пам’яттю автоматична;
- велика база документації з детальним описом методів для кожної версії та великою кількістю прикладів.

Недоліки:

- працює відносно повільно, не підтримує багатопоточність;
- велика кількість фреймворків постійно з’являється, через що їх стає занадто багато;
- неможливо слідкувати за використанням пам’яті;
- не має розвинутої групи бібліотек для створення систем машинного навчання;
- використовує великий обсяг оперативної пам’яті.

2.1.3. Python

Python – інтерпретована мова програмування високого рівня, загального призначення. Створена Гвідо ван Россумом і вперше випущена в 1991 році, філософія дизайну Python підкреслює читабельність коду завдяки помітному використанню значного пробілу. Її мовні конструкції та об'єктно-орієнтований підхід мають на меті допомогти програмістам написати чіткий логічний код для малих та масштабних проєктів [8].

Python підтримує, наприклад, такі парадигми програмування як:

- структурне;
- об’єктно-орієнтоване;
- функціональне;
- імперативне;
- аспектно-орієнтоване.

Особливості мови Python:

- динамічна типізація;
- автоматичне керування пам’яттю;
- повна інтроспекція;
- оброблення виключень;
- розвинуті структури даних високого рівня;
- підтримка більш ніж однопотокового виконання.

Інтерпретатори Python доступні для багатьох операційних систем. Глобальне співтовариство програмістів розробляє та підтримує CPython, реалізацію посилань з відкритим кодом. Некомерційна організація, програмний фонд Python, керує та спрямовує ресурси для розробки Python та CPython. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM та інших.

Розробники Python прагнуть уникнути передчасної оптимізації та відхиляти патчі до некритичних частин реалізації еталонної програми CPython, які пропонують незначне збільшення швидкості оброблення за рахунок чіткості роботи. Коли для розроблюваного проєкту важлива швидкість виконання, програміст Python може переміщувати критичні за часом функції в модулі розширення, написані мовами, такими як C, або використовувати PyPy, якщо використовуваний компілятор здатен працювати з такими модулями. Також доступний Cython, який переводить скрипт Python на C і здійснює прямі виклики API рівня C в інтерпретаторі Python.

У Python тип змінної визначається лише під час виконання, тобто мова є динаміно типізованою. Через це «присвоювання значення змінної» краще не використовувати, а замість цього вживати «зв'язуванні значення з деяким ім'ям».

Велика стандартна бібліотека функцій Python, яку зазвичай називають однією з найбільших сильних сторін мови, надає інструменти, придатні для виконання багатьох завдань. Для Інтернет-додатків підтримуються багато стандартних форматів і протоколів, таких як MIME і HTTP. Бібліотека включає в себе модулі для створення графічних інтерфейсів користувачів, підключення до реляційних баз даних, генерування псевдовипадкових чисел, засоби виконання арифметичних дій з десятковими знаками довільної точності, маніпулювання регулярними виразами та тестування одиниць.

На Python реалізована велика кількість проєктів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

Бібліотеки NumPy, SciPy і Matplotlib активно використовується замість Matlab, IDL та ін., оскільки є некомерційними, мають відкритий код; для користування ними непотрібно вивчати нову мову програмування, а також вони активно розвиваються.

Переваги:

- постійно розвивається як мова, так і її бібліотека;
- має багато бібліотек;
- має бібліотеки для машинного навчання та оброблення даних;
- код гарно читається;
- швидке розроблення прототипу;
- велика спільнота розробників, що постійно розроблюють відкриті бібліотеки;
- гарна стандартна бібліотека;
- кросплатформна;
- можливість використання парадигми функціонального програмування;
- інтерактивний режим;
- велика база з описом функцій основних бібліотек з прикладами їх використання.

Недоліки:

- відносно повільна швидкодія;
- досі поділена на версії 2 та 3, навіть через багато років після випуску версії 3;
- недостатній синтаксис для анонімних функцій;
- досить складна бібліотека з функціями багатопоточності;
- потребує багато оперативної пам’яті;

- управління пакетами є досить заплутаним.

Отже, проаналізувавши наведені особливості, переваги та недоліки зазначених вище мов програмування, для розроблення програмної системи було обрано мову Python завдяки великій кількості наявних бібліотек, а також її універсальності, кросплатформності, фремворкам для розроблення систем машинного навчання. Також мова має якісну підтримку й розвиток бібліотек й самої мови та високу швидкість розроблення прототипів.

2.2. Вибір середовища розроблення

Для середовища розроблення, виходячи зі зробленого в п.2.1 вибору мови програмування, висуваються такі вимоги:

- підтримка мови Python;
- можливість безкоштовного використання;
- розвинутість систем аналізу та рефакторингу коду;
- швидкодія;
- підтримка git.

2.2.1. Microsoft Visual Studio

Microsoft Visual Studio – це інтегроване середовище розробки (IDE) від Microsoft. Воно використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-додатків, веб-служб та мобільних додатків [9].

Visual Studio використовує платформи Microsoft для розроблення програмного забезпечення, такі як API Windows, Windows Forms, Windows Presentation Foundation, Windows Store та Microsoft Silverlight. Воно може створювати як власний код, так і керований код.

Visual Studio також надає такий редактор коду, що підтримує IntelliSense (компонент доповнення коду), а також рефакторинг коду. Інтегрований налагоджувач працює як налагоджувач на рівні початкового коду, так і налагоджувач на рівні виконання коду [10].

Інші вбудовані інструменти включають кодовий профайлер, конструктор для побудови програм GUI, веб-дизайнер, дизайнер класів та дизайнер схем бази даних. Вона містить плагіни, які розширюють набір можливостей для розроблення та тестування майже на кожному рівні.

Також середовище розроблення має додавання підтримки для систем керування джерелами (наприклад, Subversion і Git) та додавання багатьох нових наборів інструментів, таких як редактори та візуальні дизайнери для мов, що задаються доменом або набори інструментів для деяких інших аспектів розробки програм життєвого циклу.

Переваги:

- IntelliSense – зручний інтерфейс коригування та рефакторингу коду;
- розвинуте середовище розроблення та debugger;
- наявність великої кількості плагінів;
- наявність детальної документації;
- швидкість виконання функцій;
- проста для використання;
- відсутність помилок.

Недоліки:

- громіздке, оскільки включає багато функцій, для багатьох з яких необхідне своє меню, вікно чи команда на клавіатурі;
- безкоштовна версія обмежена у можливостях відносно повної, а повні версії дорогі та не мають студентської безкоштовної підписки.

2.2.2. Visual Studio Code

Розроблений Microsoft для Windows Linux та ОС, VS Code є розширюваним редактором коду, який не слід плутати з Visual Studio [9]. Дійсно VS Code невеликий, але повний, і програмне забезпечення є відкритим кодом за ліцензією MIT, саме це характеризує основну різницю

між Visual Studio та VS Code. Перша версія VS Code була опублікована 29 квітня 2015 року. VS Code порівнювався з Atom за набором можливостей. Дійсно, як і Atom, VS Code побудований на Electron, а це означає, що ці дві програми мають майже однакові переваги та недоліки [11].VS Code – це проєкт Microsoft, який має найбільшу кількість учасників GitHub. Ця ініціатива підвищила славу Microsoft і позиціонувала її як одного з головних гравців спільноти з розроблення програмного забезпечення. У цьому випадку можливо додати нову мову до середовища, наприклад, Python. Достатньо завантажити та встановити відповідний плагін, щоб адаптувати його до навколишнього середовища. VS Code удосконалений такими функціями, як інтеграція потужного двигуна автоматичного завершення коду (IntelliSense), консолі налагодження та терміналу для запуску серверних команд.

Головною перевагою VS Code є те, що він має архітектуру середовища на основі розширень, а оскільки IDE легкий, його можна розширити, додавши необхідні компоненти за потребою.

Переваги:

- більше 4700 розширень;
- являє собою потужний інструмент управління кодом;
- швидкий;
- підтримує багато мов;
- імпорт клавіатурних скорочень з інших редакторів Python, таких як Sublime Text або Atom.

Недоліки:

- VS Code не дуже підходить для оброблення великих файлів коду;
- складно знайти розширення, яке найкраще відповідає потребам розробників через тисячі доступних розширень;
- статичний аналіз коду не є розвиненим як у повноцінних середовищ розроблення.

2.2.3. PyCharm

PyCharm – один з широко використовуваних Python IDE, який був створений Jet Brains. Це одна з найрозвинутіших IDE для Python.

Він відрізняється від конкурентів завдяки своїм інструментам підвищення продуктивності роботи розробника, таким як «швидкі виправлення». Доступний у трьох версіях: у Community версії з ліцензією Apache, у навчальній (Edu) версії та у власній версії Professional. Перші дві версії є відкритим кодом і тому є безкоштовними, тоді як версія Professional не є безкоштовною.

Community версія спільноти є дуже цікавою, оскільки має різні функції, такі як підсвічування синтаксису, автоматичне доповнення та перевірка коду в реальному часі. Платна версія, очевидно, має більш вдосконалені функції, такі як повне управління базами даних і підтримку безлічі важливих фреймворків, таких як Django, Flask, Google App, Engine, Pyramid та web2py.

Особливості:

- оснащення інтелектуальним редактором коду, розумною навігацією по коду, швидким та безпечним рефакторингом;
- інтегрованість з такими функціями як налагодження, тестування, профілювання, розгортання, віддалене розроблення та інструменти бази даних;
- підтримка фреймворків веб-розробки JavaScript, HTML, CSS, Angular JS та Live edit;
- підтримування інтеграції з IPython, консоллю python та науковими бібліотеками для роботи з даними.

Переваги:

- надає розвинуту платформу розробникам, яка допомагає їм при автоматичному заповненні коду, виявленні помилок, швидкому їх виправленні тощо;

- надає підтримку для багатьох фреймворків за рахунок збільшення великої кількості факторів економії ресурсів;
- підтримує таку функцію, як кросплатформне розроблення, щоб розробники могли писати код для різних платформ;
- має функцію налаштуваня інтерфейсу, що, в свою чергу, збільшує продуктивність розроблення;
- має безкоштовну повну версію для студентів.

Недоліки:

- PyCharm – це дорогий інструмент, враховуючи особливості та інструменти, які він надає клієнту;
- початкове налаштування важке і іноді може зависати.

Беручи до уваги дані характеристики середовищ розроблення, було прийнято рішення обрати PyCharm, як середовище розроблення з найбільшою інтеграцію з мовою Python (на відміну від Visual Studio) та розвинутими засобами аналізу та рефакторингу коду (на відміну таких від VS Code).

3. СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ СИСТЕМИ

3.1. Загальна організація системи

Система реалізована у вигляді набору основних модулів, які утворюють ядро, бази даних та Telegram-боту у якості інтерфейсу користувача (рис. 1). Один з модулів містить у собі підмодулі для збирання відгуків та оцінок з різних джерел. На зображенні вони умовно названі модулі 1, 2, …, N. Кожен з модулів відповідає за окремий ресурс у мережі Інтернет. При цьому, дані, що повертає модуль, залежать від джерела, яке він оброблює.

Власне, сама розороблена програмна система має такі модулі:

- модуль роботи з Telegram-ботом;
- модуль визначення емоційної оцінки відгуку;
- модуль передоброки тексту;
- модуль навчання моделі аналізу емоцій відгуку;
- модуль збереження та завантаження нейронної мережі аналізу емоцій відгуку;
- загальний модуль збирання відгуків;
- підмодулі збирання даних про відгуки з джерел відгуків;
- модуль роботи з базою даних;
- модуль аналізу зібраних даних.

Дана структура була розроблена, оскільки вона дозволяє швидко проєктувати систему і, відповідно, швидко її розроблювати, що, у свою чергу, надає можливість створити мінімально працюючу версію програми у короткий термін часу. Також варто зазначити, що дана структурна організація системи робить можливим додавання нових модулів для оброблення джерел відгуків зі змінами лише в одному модулі.

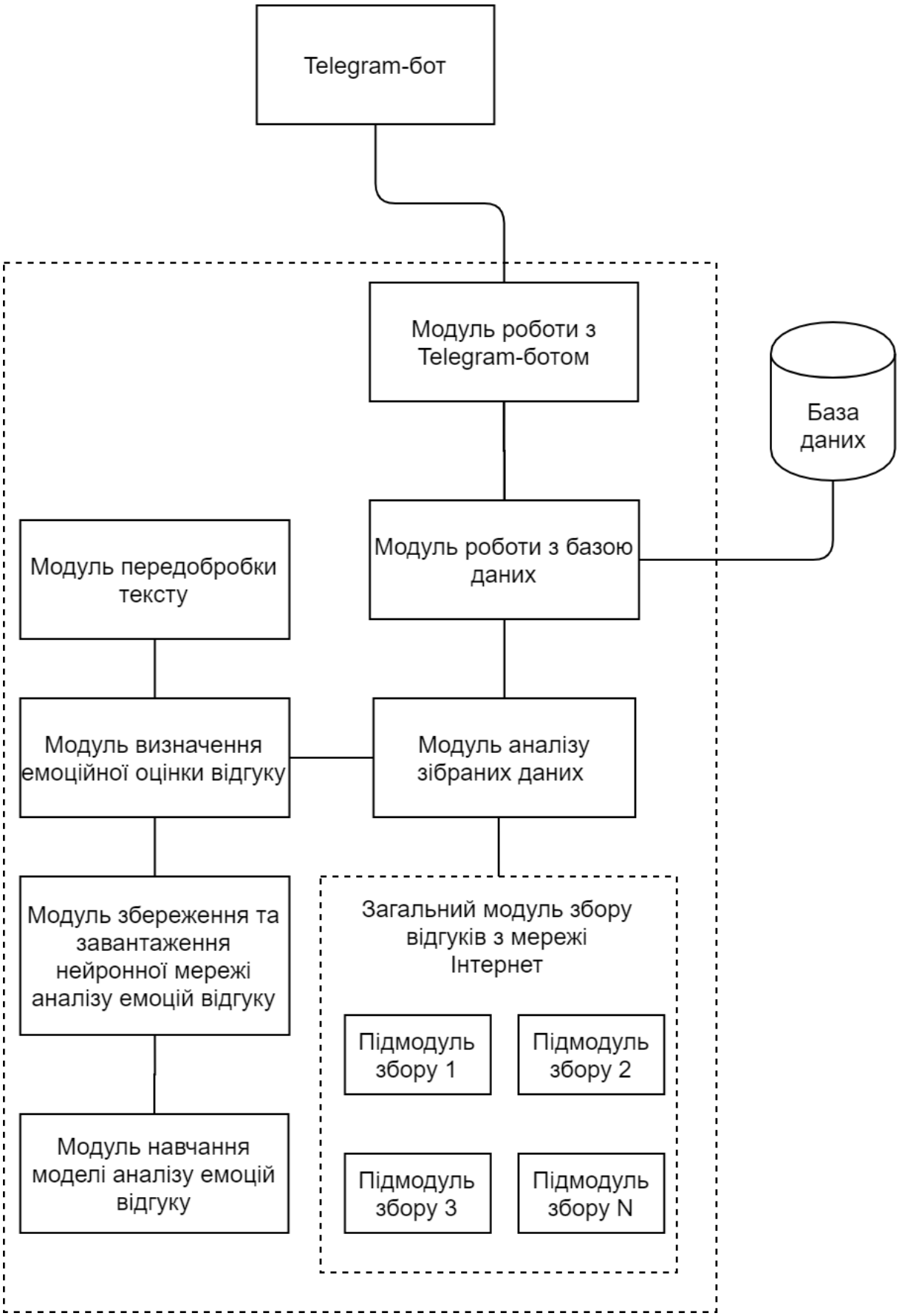


Рис. 1. Структурна організація розробленої системи

Також, однією з переваг даної організації є можливість швидко розбити дані модулі на підсистеми з можливістю перетворити наявну систему на систему з мікросервісною архітектурою без створення модулів з нуля [12]. Перевагами мікросервісної архітектури у даному випадку буде можливість підвищити швидкодію системи за рахунок роботи багатьох частин системи абсолютно незалежно одна від одної. Наприклад, таким окремим мікросервісом може стати сервер Telegram боту, який буде лише приймати та відправляти дані користувачам. До того ж, разом з цим з'явиться можливість розроблення нових користувацьких інтерфейсів та окремих серверів для роботи з ними.

Запропонована в даному дипломному проєкті система організації модулів орієнтована на мінімальну зв'язність модулів та їх сильне зціплення. Завдяки цьому підтримується якісне розділення на модулі та можливість їх подальшого розділення для використання у інших системах чи підсистемах.

У якості інтерфейсу користувача був обраний Telegram-бот завдяки набору переваг даного сервісу. Кількість користувачів цього месенджеру перебільшує чотириста мільйонів і при цьому більшість з них звикла до інтерфейсу й користуванням ботами, що потенційно може полегшити розповсюдження бота серед користувачів у цьому сервісі.

Також важливою перевагою Telegram-боту є простота розроблення інтерфейсу, оскільки потрібно оброблювати лише текстові команди, а відправлення й отримання даних покладена на Telegram. При цьому автоматично отримується підтримка на багатьох платформах, оскільки Telegram має свої додатки для майже всіх актуальних на наш час середовищ систем: Android, Windows, iOS, Linux та веб. Це дозволяє користуватися їм на будь-якій системі, що має можливість виходу в мережу Інтернет за допомогою сучасних браузерів.

3.1.1. Модуль роботи з телеграм ботом

Модуль роботи з телеграм ботом отримує запити від користувача та, в залежності від типу команди, оброблює їх. Також, використовуючи модуль аналізу зібраних даних, відправляє користувачу результати роботи модуля аналізу відгуків. Для роботи з API Telegram використовується відкрита бібліотека python-telegram-bot [14], що є обгорткою API Telegram [13]. Дана бібліотека обрана, виходячи з того, що вона є найбільш розвинутою, має найбільше користувачів та постійно оновлюється. Варто зазначити, що робота з API телеграму без використання обраної бібліотеки вимагала би в рази більше часу для реалізації.

Набір команд, що оброблює модуль, є досить компактним. Завдяки цьому інтерфейс користувача є досить простим та не вимагає багато часу для вивчення. Ще однією перевагою такого інтерфейсу є знижена можливість вводу некоректних даних, що спрощує перевірку вводу на правильність. До списку команд входять такі команди:

- «help»;
- «links»;
- «add»;
- «analyze»;
- «hist»
- «remove».

3.1.2. Модуль визначення емоційної оцінки відгуку

Модуль отримує на вхід текст відгуку, що вже оброблений модулем передоброблення тексту, що полегшує вирішення задачі оцінювання тональності тексту. На виході отримуємо оцінку за системою PERMA (рис. 2) у десяти вимірах: позитивні емоції, залучення, відносини, значення, завершеність, кожен з яких має відповідно позитивну та негативну оцінку [15]. Така система оцінювання була обрана з огляду на можливе майбутнє використання для більш точного оцінювання ніж позитивний чи негативний

відгук. Наприклад, завдяки даній системі оцінювання можливо знайти найбільш схожий за емоціями відгук від іншого користувача.

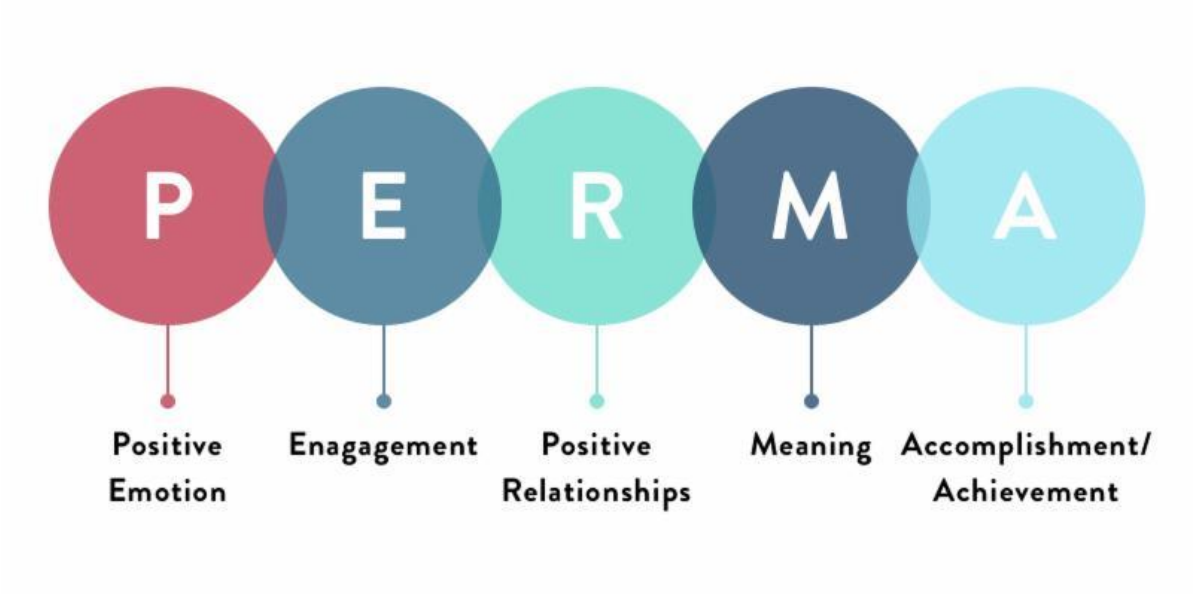


Рис. 2. Система PERMA

Модуль базується на використанні функцій бібліотеки Keras [16]. Keras містить численні реалізації часто використовуваних будівельних блоків нейронних мереж, таких як шари, цілі, функції активації, оптимізатори та багато інструментів для полегшення роботи з зображеннями та текстовими даними, щоб спростити написання коду глибокої нейронної мережі. Окрім стандартних нейронних мереж, Keras має підтримку згорткових і рекурентних нейронних мереж. Це також дозволяє використовувати розподілене навчання моделей глибокого навчання на кластерах одиниць графічної обробки (GPU) та тензорних процесорів (TPU), головним чином спільно з CUDA [17].

3.1.3. Модуль передобробки тексту

Модуль отримує на вхід текст відгуку, що отриманий напряму з джерела. Такий відгук містить у собі всі слова: як значущі для подальшої

оцінки емоцій, так і шум. Однією із задач даного модулю, власне, є видалення цих незначущих слів.

Такі слова прийнято називати «стоп-словами» [18]. Хоча стоп-слова, зазвичай, стосуються найпоширеніших слів у мові, не існує єдиного універсального списку стоп-слів, використовуваних усіма засобами оброблення природньої мови. Будь-яку групу слів можна вибрати як стоп-слова відповідно до заданої мети. Для деяких пошукових систем це декілька найпоширеніших коротких функціональних слів, таких як «є», «на», «який» і так далі. У цьому випадку стоп-слова можуть спричинити проблеми під час пошуку фраз, що включають їх, особливо в назвах зі стоп-слів, наприклад таких як «The Who», «The», or «Take That». Інші пошукові системи видаляють деякі найпоширеніші слова, включаючи лексичні слова, наприклад «хочу», із запиту з метою підвищення продуктивності. Для вилучення таких слів обрана бібліотека NLTK [19], оскільки вона є розвинутим засобом для оброблення природньої мови багатьма способами для роботи з текстом. Саме відповідно до її набору стоп слів і виконується прибирання шуму у розробленій системі.

Перед прибиранням шумів спочатку треба виконати токенізацію [20]. Токенізація – перетворення послідовності символів у послідовність токенів. Токенізація також виконується засобами бібліотеки NLTK. У цьому модулі токени являють собою слова.

Для спрощення визначення емоцій використовується стемінг – процес приведення перетворених (або іноді похідних) слів до їх основи або кореневої форми – як правило, письмової словоформи [21]. Стема не повинно бути тотожним морфологічному кореню слова; зазвичай достатньо, щоб споріднені слова відображалися на одній стемі, навіть якщо ця стема сама по собі не є коректним коренем. Стемінг також виконується за допомогою засобів бібліотеки NLTK. Текст після стемінгу залишає лише основи слів, які простіше аналізувати потім. Варто зазначити, що не у всіх системах використовується стемінг. Також інколи використовується

лематизація – заміна слова на його синонім. Інколи нічого з цього не виконується. Вибір залежить від побудови системи. Для використовуваних у системі алгоритмів було вирішено використовувати дані засоби.

3.1.4. Модуль навчання моделі визначення тональності відгуку

Модуль призначений для навчання моделі та її збереження за допомогою модуля збереження та завантаження нейронної мережі Keras[22].

Навчання нейронної мережі – це адаптація мережі для кращого вирішення завдання шляхом розгляду вибіркових спостережень. Навчання включає коригування вагів у нейронах мережі для підвищення точності результату. Це робиться шляхом мінімізації спостережуваних помилок. Навчання закінчено, коли вивчення додаткових спостережень негативно знижує рівень помилок. Навіть після навчання частота помилок зазвичай не дорівнює 0. Якщо після навчання рівень помилок занадто високий, мережу зазвичай потрібно переробити. Практично це робиться шляхом визначення функції значень, яка періодично оцінює результати під час навчання. Поки результат функції продовжує скорочуватись, навчання продовжується. Вартість часто визначається як статистика, значення якої може бути лише приблизним. Навчання намагається зменшити загальну різницю в результатах між спостереженнями.

Оскільки навчання моделі зазвичай є досить ресурсозатратним процесом, розглядається можливим навчати модель на платформі для запуску коду для нейронних мереж Google Colab [23], що безкоштовно дає доступ до виконання коду на досить потужних GPU. Даний підхід розвантажує обладнання, на якому проводиться розроблення системи, та пришвидшує сам процес навчання у декілька разів.

Для навчання використовується набір даних із слів та простих словосполучень, для яких поставлені оцінки за системою PERMA. Набір даних являє собою дещо модифікований для навчання набір даних PERMA.

3.1.5. Модуль збереження та завантаження нейронної мережі аналізу емоцій відгуку

Даний модуль зберігає після навчання та завантажує при старті системи нейронну мережу. Збереження та завантаження відбувається стандартними засобами Keras для збереження таких даних мережі, як її будова, налаштування та значення змінних у кожному нейроні, що були отримані під час навчання.

Нейронна мережа зберігається у двох файлах:

- файл з даними про шари(їх кількість та кількість нейронів у шарах) мережі у форматі JSON або XML;
- файл з даними про коефіцієнти нейронів у шарах мережі у форматі HDF5 [24].

3.1.6. Загальний модуль збору відгуків з мережі Інтернет

Модуль відповідає за збирання відгуків з мережі Інтернет. Для збору він використовує підмодулі, кожен з яких збирає дані з одного сайту. Кожен підмодуль є незалежним. До того ж, дані, що повертає модуль, залежать від реалізації модулю та початкового вмісту веб-сторінок. Наприклад, з деяких ресурсів може бути надзвичайно важко зібрати текстові відгуки, а деякі ресурси таких відгуків можуть взагалі не мати. Тому підмодулі опціонально повертають цифрові оцінки та/або текстові відгуки.

Одним із способів збирання даних для підмодуля є веб-скрапінг [25]. Веб-скрапінг – це спосіб збирання даних з веб-сайтів. Програмне забезпечення для скрапінгу веб-сторінок може отримати доступ до всесвітньої павутини безпосередньо за допомогою протоколу передачі гіпертексту або через веб-браузер. Хоча веб-скрапінг може бути здійснений вручну користувачем програмного забезпечення, термін зазвичай стосується автоматизованих процесів, реалізованих за допомогою бота або веб-сканера. Це форма копіювання, в якій конкретні дані збираються та

копіюються з Інтернету, як правило, в центральну локальну базу даних або електронну таблицю для подальшого пошуку або аналізу.

Розшифровка веб-сторінки включає в себе її вилучення та вилучення з неї даних. Фетчинг – це завантаження сторінки (що робить браузер під час перегляду сторінки). Тому сканування веб-сторінки є головним компонентом скрапінгу веб-сторінок для отримання сторінок для подальшої обробки. Після отримання вмісту веб-сайту можливо продовжити подальший пошук даних. Вміст сторінки може бути проаналізований, знайдений, переформатований, його дані скопійовані в електронну таблицю тощо. Веб-скрапери, зазвичай, виймають щось зі сторінки, щоб використовувати дані з іншою метою десь в іншому місці. Прикладом може бути пошук та копіювання імен та номерів телефонів або компаній та їх URL-адрес до списку.

Іншим способом збору даних може бути використання публічного API джерела або платного API для певного конкретного ресурсу.

3.1.7. Модуль роботи з базою даних

Модуль забезпечує роботу з базою даних. Для цього у ньому зберігаються посилання та ключі доступу до бази даних. Вся взаємодія з базою даних має проходити через цей модуль. Таким чином імплементується шаблон проектування «Фасад» – шаблон, який зазвичай використовується в об'єктно-орієнтованому програмуванні. Аналогічно фасаду в архітектурі, фасад в програмуванні – це об'єкт, який виконує функцію переднього інтерфейсу, маскуючи складніший базовий або структурний код. «Фасад» був обраний, оскільки він може:

- покращити читабельність та зручність використання модулю програмної системи шляхом маскування взаємодії зі складнішими компонентами за одним (і часто спрощеним) API;
- забезпечити інтерфейс для більш загальних можливостей (у комплекті з валідацією введення, що залежить від контексту);

- служити точкою старту для більш широкого рефакторингу монолітних або щільно з'єднаних систем на користь більш вільно зв'язаного коду.

3.1.8. Модуль аналізу зібраних даних

Модуль аналізує дані у вигляді цифрових оцінок та результатів аналізу емоційного забарвлення відгуків. Для аналізу На виході отримуємо текстові результати аналізу отриманих даних.

3.2. Узагальнений алгоритм роботи системи

Даний алгоритм відповідає за основу роботи програми: отримання вводу, оброблення команди тим чи іншим способом та відправка результату виконання користувачу. Схематично алгоритм роботи та взаємодії модулів системи зазначений на (рис. 3).

Алгоритм складається з таких етапів:

1. Отримання вводу від користувача. За даний пункт відповідає модуль роботи з телеграм ботом. Користувач вводить ту чи іншу команду з параметрами для деяких команд. Модуль отримує ці дані разом з інформацією про користувача. Наприклад, ім’я та прізвище, ідентифікатор чату, ідентифікатор користувача та інші дані. Найбільш необхідним є ідентифікатор чату, оскільки саме за ним буде відправлено відповідь користувачу.
2. Перевірка введених даних на коректність. Ця частина алгоритму також реалізована у модулі взаємодії з Telegram ботом. У тому випадку, якщо введені дані некоректні, користувач отримає повідомлення про помилку.
3. Модифікація списків посилань для пошуку відгуку. В залежності від отриманих від користувача запитів, редагуються записи у базі даних про список ресурсів, з яких необхідно збирати інформацію. Також реалізовано у модулі взаємодії з Telegram ботом.

4. Очікування старту пошуку відгуків, оскільки система вже може бути завантажена пошуком або ж час періоду нового пошуку ще не настав. Черга виконання задається у стартовому методу програми.
5. Пошук відгуків. Реалізовано у загальному модулі збору відгуків з мережі Інтернет. Збір з кожного окремого ресурсу виконується у підмодулях збору відгуків.
6. Аналіз тональності відгуків – реалізовано у модулі передобробки та, власне, модулі аналізу тексту. На виході отримуємо результати аналізу емоцій тексту.
7. Аналіз отриманих результатів – реалізовано у модулі аналізу. У даному модулі відбувається формування звіту про результати дослідження оцінок та відгуків. Саме ці дані будуть використані для відправлення користувачу.
8. Збереження результатів. Для можливого використання у аналізі історії з порівнянням результатів аналізу за певний час або просто порівняння з минулим результатом, відбувається збереження даних, отриманих після стадії аналізу. Зберігання відбувається за допомогою модулю взаємодії з базою даних.
9. Відправлення результатів аналізу користувачу. За допомогою модуля взаємодії з Telegram ботом відправляється результат аналізу відгуків користувачу.

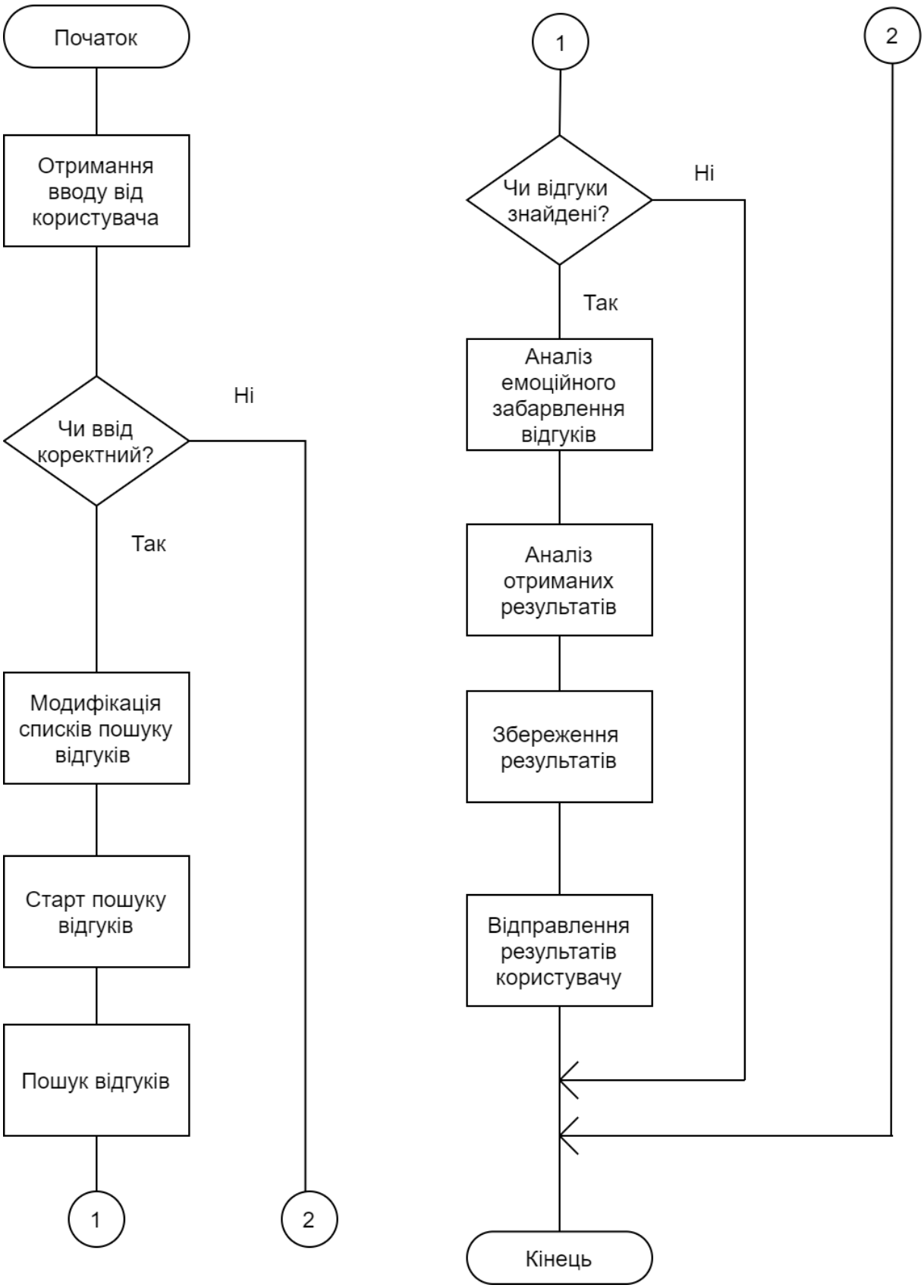


Рис. 3. Узагальнений алгоритм роботи системи

3.3. Узагальнений алгоритм аналізу текстових відгуків

Алгоритмом передбачено виконання передобробення та власне аналізу тональності тексту. Схема алгоритму зображена на (рис. 4).

Алгоритм виконується у такому порядку:

1. Токенізація - відбувається розбиття початкового тексту на токени для подальшого оброблення. У даному випадку токени є окремими словами. Токенізація відбувається у модулі передоброблення тексту.
2. Прибирання стоп-слів - реалізоване у модулі передобролення тексту. Зі списку токенів прибираються стоп-слова.
3. Стемінг – також проходить у модулі передоброблення тексту. Послідовність токенів оброблюється таким чином, щоб перетворити слова, утворені з інших слів, на слова, з яких вони були утворені. Таким чином можливе покращення точності аналізу емоцій.
4. Аналіз емоційного забарвлення за моделлю PERMA - реалізований у модулі аналізу емоційного забарвлення тексту. Тут за допомогою нейронної мережі отримується оцінка емоційного забарвлення тексту за п’ятьма параметрами, кожен з яких має позитивний та негативний вимір.
5. Аналіз отриманих оцінок. У модулі аналізу проходить обчислення оцінки відгуків на основі оцінки емоційного забарвлення тексту. Завдяки оцінці за п’ятьма параметрами у подальшому можливо розширити аналіз або покращити якість аналізу.

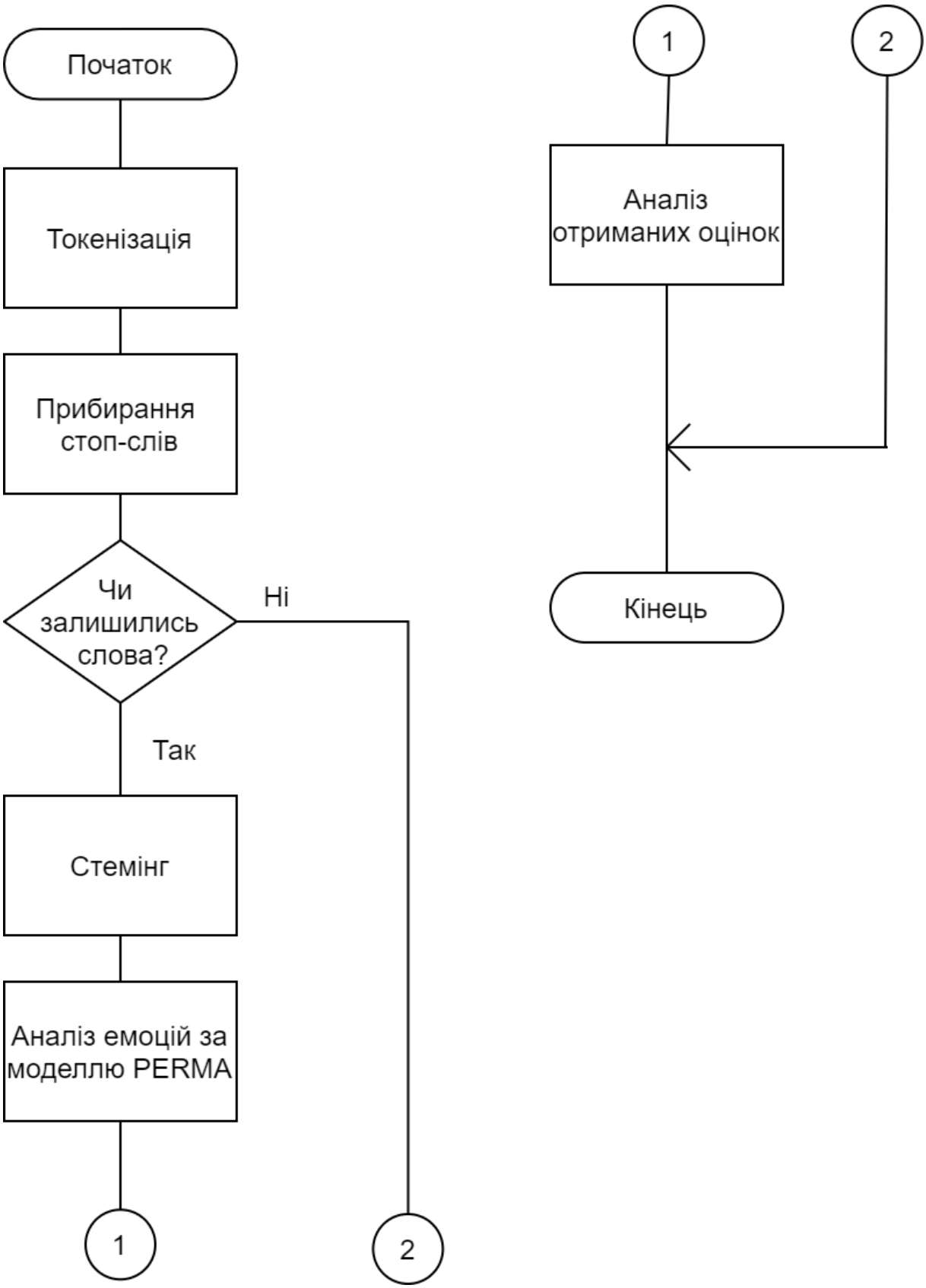


Рис. 4. Алгоритм аналізу текстових відгуків

4. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості реалізації нейронної мережі аналізу текстових відгуків за системою PERMA

Модель реалізована з використанням звичайних шарів dense та шарів dropout.

Dense шар являє собою шар у якому відбувається множення матричного вектора. Значення в матриці шару – це відстежувані параметри, які оновлюються під час зворотного поширення. Таким чином отримується *m*-розмірний вектор як вихід. При цьому dense шар використовується для зміни розмірів вектора. Він застосовує обертання, масштабування, перетворення до вектора [26].

Dropout – це патентована Google технологія регуляризації для зменшення перенавчання в нейронних мережах шляхом запобігання складним спільним адаптаціям даних про навчання [27]. Це ефективний спосіб виконання усереднення моделей з такими нейронними мережами. Термін «dropout» відноситься до випадання одиниць (як прихованих, так і видимих) в нейронній мережі.

Функція активації, що використовується у всіх шарах – sigmoid [28].

Для створення embeddings використовується BERT [29]. Це методика попередньої підготовки NLP, розроблена Google. BERT був створений та опублікований у 2018 році. Причини високої продуктивності BERT у завданнях з розуміння природних мов ще недостатньо вивчені. Поточні дослідження були зосереджені на дослідженні взаємозв'язку виходу BERT в результаті ретельно підібраних вхідних послідовностей, аналіз внутрішніх векторних уявлень за допомогою зондуючих класифікаторів та зв’язків, представлених вагами уваги. За допомогою бібліотеки sentence transformers, що побудована на BERT, отримуються 768-мірні embeddings, які потім використовуються для навчання нейронної мережі [30].

Для навчання моделі використовується набір даних, у якому для слова наведена відповідна йому оцінка за PERMA. Для навчання було проведено попередню обробку оригінального набору даних [31]. У ньому кожному слову відповідала одна оцінка, при цьому слово могло дублюватися. Кожен раз оцінювався лише один з цих параметрів. Суть такого попереднього оброблення датасету полягала у приведенні датасету у форму, у якій у одному рядку наведене слово має всі свої оцінки, а у разі, якщо оцінка за якимсь з параметрів відсутня, то вона встановлювалася рівною нулю.

Також, набір даних був очищений від наборів знаків, що представляли собою смайлики, пошкоджених даних та деяких інших варіантів, що могли ускладнити навчання моделі. При цьому кількість унікальних слів або словосполучень скоротилася приблизно на 800 – 1000, що складає майже одну шосту від початкового датасету. Дані операції не завадять оцінці відгуків Інтернет-користувачів, оскільки згідно з проведеним дослідженням, використання деяких прибраних наборів символів у відгуках майже взагалі не відбувається. Інші ж набори символів, якщо і зустрічаються, то з набагато меншою ймовірністю ніж у початковому наборі даних, у якому їхня кількість наближається до 20% від усієї кількості слів.

У результаті була отримана модель, що може достатньо показово використовуватися для оцінки окремих слів та невеликих словосполучень за системою PERMA. З особливостей варто зазначити, що використання моделі потребує попереднього отримання embeddings, що є досить повільною операцією. Через специфіку датасету, модель найкраще працює на досить сильно забарвлених словах, при цьому увагу необхідно звертати саме на найбільші оцінки. Оскільки датасет не повністю оцінює слово та виходячи з того факту, що оцінювання слів у ньому є неповним, результати для окремих слів, що мають ненульові оцінки у всіх емоціях потрібно аналізувати в подальшому з обережністю.

Завдяки embeddings BERT вдалося побороти обмеження у вгляді невеликої кількості слів у початковому датасеті, оскільки завдяки embeddings також можливо отримати й оцінки для близьких за вектором слів. Однак, оскільки кількість слів у наборі даних все ще є малою та те, що деякі слова можуть повторюватися у фразах, наприклад «baby» та «a baby», далеко не всі слова отримають оцінку, близьку до реальної. Через це інтегральна оцінка тексту більш наближена до реальної оцінки.

При цьому така оцінка майже не може наблизитися до цифрових результатів близьких до 0.5 за хоча б одним із показників на реальному тексті через те, що у таких текстах використовуються слова різної забарвленості. Відповідно, цю особливість було враховано в подальшому у модулі аналізу для нормальної роботи, завдяки чому досягнута нормальна роботи системи (рис. 5)

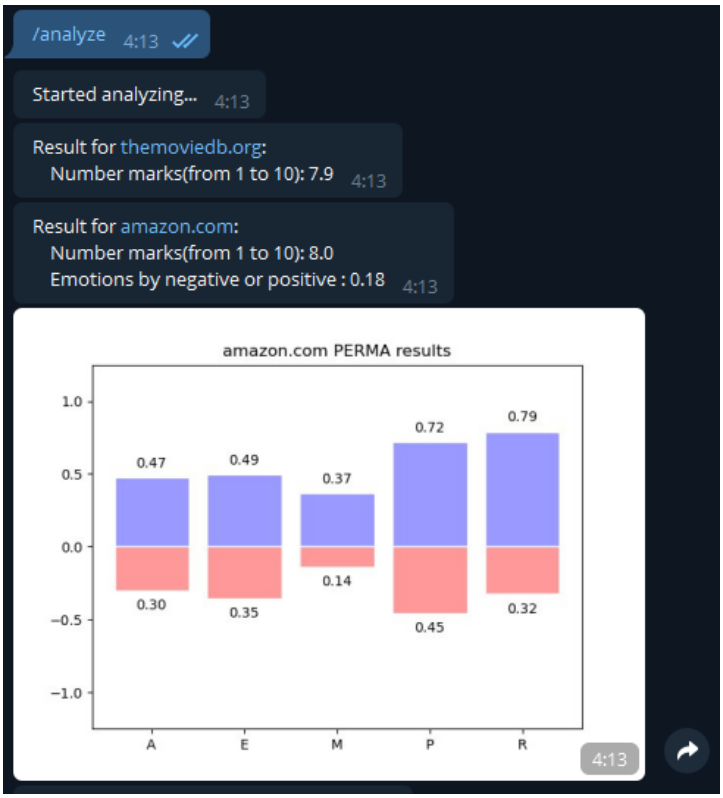


Рис. 5. Приклад роботи системи

4.2. Тестування системи

Система була протестована за методом «білої скриньки», тобто беручи до уваги код програми та його найбільш ймовірні слабкі місця.

Перший етап аналізу здійснювався автоматично завдяки обраному середовищу розроблення PyCharm, яке має статичний аналізатор коду. Завдяки цьому найбільш ймовірні помилки з використанням неіснуючих змінних, неправильні виклики функцій та ін. були перевірені під час створення системи.

Наступними тестами були перевірки роботи модулів збирання даних. Для кожного з п’яти модулів створено по два тести, що перевіряють наявні методи у класах модулів збору: метод перевірки посилання на приналежність до ресурсу, що оброблює даний збирач, та власне метод збору даних. Також перевірений загальний модуль збору на наявність у ньому всіх збирачів та правильний вибір конкретного модулю для збору даних.

Тести моделі аналізу емоцій проводилися при навчанні моделі автоматично та після навчання. Перевірки при навчанні здійснювались як за допомогою розділення набору даних на тестовий набір та набір для навчання, так і без цього. Навчання моделі при великій кількості епох наближалася до досить значної, тобто більше 70%. Покази тестового набору завжди коливалися близько до 15%. Ручні тести системи показали непристосованість даної моделі до тестування, оскільки найкращі результати отримувалися на приблизно 10 епохах навчання. При такій кількості епох отримувані результати були найбільш репрезентативними. Збільшення кількості епох приводило до стрімкого погіршення результатів тестів через перенавчання, не дивлячись на шари dropout. Тести модулю емоційного оцінювання показали досить точні результати після впровадження декількох нормалізацій результатів, спрямованих на покращення роботи з моделлю.

4.3. Реалізація модулів збору даних

Модулі збору даних реалізовані за допомогою різних систем. Три з них використовують спеціальні бібліотеку, що адаптує API ресурсів. Такими ресурсами є OMDB та TMDB.

Збір даних з IMDB також реалізований за допомогою бібліотеки, проте API не використовується. Замість цього бібліотека збирає дані шляхом парсингу. Оскільки найбільш важливий для даної роботи модуль бібліотеки, що збирає текстові відгуки, зараз має лише часткові функціональні можливості від потрібних для збору всіх відгуків, то збирається лише обмежена їх кількість.

Цей недолік спричинений тим, що IMDB намагається ускладнити процес збирання цих даних. Власне збір відгуків ускладнений відсутністю пагінації сторінок, замість якої є лише динамічне завантаження при натисканні відповідної кнопки на сайті. Використання ж API цього ресурсу можливе лише за окремим погодженням від адміністраторів за спеціальним зверненням та на комерційних основах.

З Metacritic дані збираються за допомогою Beautiful Soup. Це бібліотека Python для розбору HTML і XML-документів (у тому числі з неправильною розміткою, тобто незакритими тегами). Він створює дерево розбору для аналізу сторінок, які можуть бути використані для отримання даних з HTML, що корисно для скрепінгу веб-сторінок.

Завдяки наявності пагінації, з даного ресурсу все ще можливо збирати всі відгуки. Проблемою при роботі з цим сервісом є різниця у його частинах. Оскільки зараз система для показу роботу орієнтована на збір даних про фільми, то відповідно і цей модуль був розроблений відповідно до специфіки сторінок з відгуками про фільми. При цьому, наприклад, частина ресурсу з відгуками про ігри, не може бути аналізована даним модулем не дивлячись на те, що різниці у візуальному оформленні майже немає.

Amazon має ті самі обмеження на автоматизований збір, що й IMDB, тобто відгуки завантажуються при натисканні на кнопку для завантаження

нової порції відгуків (відсутня пагінація) та доступ до API є лише за окремими зверненнями до адміністрації. На додаток до цього, як і Metacritic, є різниця між однаковими на вигляд модулями. Наприклад, сторінка про комп'ютерну техніку та фільм має різні реалізації показу оцінки у зірочках. Як зазначалося у частині про модуль збору з Metacritic, система налаштована зараз на збір відгуків про фільми, тому і з цього сервісу гарантоване збирання невеликої частини відгуків зі сторінок про фільми.

На відміну від модулю для Metacritic, у цьому модулі збирання відгуків реалізоване засобами бібліотеки Selectorlib. Це комбінація двох пакетів. Розширення для браузеру Google Chrome, яке дозволяє розмічати дані на веб-сайтах та експортувати з ним файл з налаштуваннями YAML. Друга частина – бібліотека Python, яка читає цей файл YAML і витягує дані, які ви позначено на сторінці [32].

Дана бібліотека була обрана завдяки швидкості та, відповідно, простоті розмітки. Швидкість роботи самого модулю збору не бралася до уваги через те, що необхідно завантажити та опрацювати досить малий об'єм даних. Також, при спробі зробити асинхронне оцінювання для даних, зібраних завдяки цьому модулю, був виявлений той факт, що Amazon має захист від збору даних ботами у вигляді обмеження кількості запитів. Для відновлення доступу до сайту довелося використати VPN TunnelBear для тестів та зміну даних про браузер для доступу без VPN.

4.4. Напрямки подальших досліджень

Однією з проблем системи є недовершеність системи емоційного забарвлення, що спричинено багатьма факторами. Серед таких основними є особливості набору даних для навчання та ймовірного неоптимального вибору будови нейронної мережі або методики навчання.

Однією з особливостей набору даних є розподіл оцінок емоційного забарвлення за силою.

Нижче наведені додані абсолютні суми оцінок слів з оригінального набору даних:

- NEG_A: 32.93;
- NEG_E: 29.47;
- NEG_M: 30.96;
- NEG_P: 67.82;
- NEG_R: 55.78;
- POS_A: 51.68;
- POS_E: 66.94;
- POS_M: 73.95;
- POS_P: 76.05;
- POS_R: 77.83.

Neg відповідає за негативне забарвлення за емоцією, pos відповідно за позитивне. P, e, r, m, a – літери, що позначають емоції системи PERMA. Як нескладно побачити, розподіл нерівномірний. R та p мають вищі як негативні, так і позитивні оцінки. При цьому вони ж є єдиними сильними негативними оцінками. Інші негативні оцінки приблизно у 2 рази нижчі.

Даний розподіл є найбільш вірогідною причиною того, що нейтральні слова найчастіше забарвлені трохи у позитивний бік. Даний ефект вимагає своєї обробки у вигляді введенні рівня найменшої оцінки слова, при якому воно буде вважатися емоційно забарвленим та враховуватися при подальшій оцінці. Варто зазначити, що сума оцінок окремих слів, відфільтрованих за такою межею, має точну оцінку.

Власне, покращення точності даної частини системи є одним із ймовірних напрямків для подальшої роботи. Робота може вестись у напрямках проблем, зазначених вище. Тобто, першим варіантом є збір більшого та точнішого набору даних для навчання моделі.

Іншим напрямком роботи може бути водсконалення самого процесу навчання. Можлива зміна бібліотеки, шарів, будови мережі або інших параметрів може покращити роботу моделі. Також можлива набагато

більша зміна принципу роботи моделі. Реалізована система не бере до уваги сусідні слова при оцінюванні. Можливе застосування алгоритмів оцінки, що беруть до уваги послідовність слів та оцінюють лише позитивне та негативне забарвлення з їх адаптацією до оцінки за більшою кількістю параметрів.

Ще одним напрямком для розвитку є багатомовність. Одним із варіантів отримання системи з можливістю оцінки текстів на багатьох мовах та теоретичним низьким затратам на зміну системи, є використання багатомовних embeddings. Таким чином, використовуючи датасет лише, наприклад, на англійський мові, за допомогою embeddings можна аналізуючи слово на іншій мові, знайти близький йому вектор на англійській.

Також, отримання системи, яка зможе кодувати у embeddings одразу весь текст, а не його окремі частини, значно пришвидшить роботу системи. Модель у даній реалізації при кодуванні усього тексту та подальшій оцінці моделлю завжди отримує результати, у яких жодна з емоцій не переважає. Можливо, такі результати покращяться автоматично при зміні датасету. Іншим варіантом розвитку є прибирання нейтральних слів перед кодуванням.

Додатковими логічними напрямками розвитку системи є додавання інших варіантів інтерфейсу, відсіювання підроблених відгуків та більш глибокий аналіз даних, а саме аналіз змін результатів у часі. Підроблені відгуки є проблемою у двох випадках: підроблені на замовлення та відгуки у вигляді хвилі, спричинені масовими оцінками від користувачів у якості протесту на певну подію, у якості соціального явища або інших причин. Наприклад, це може бути потік негативних відгуків на фільм, актор у якому здійснив якийсь неприйнятний для частини аудиторії вчинок. Такі відгуки не оцінюють сам продукт, через що оцінка останнього викривляється. Для подолання таких відгуків їх можливо фільтрувати. Схожа система

фільтрування нещодавно була введена у магазині Steam, тобто реалізація такої фільтрації точно є можливою.

ВИСНОВКИ

Метою даного дипломного проєкту є розроблення програмної системи для визначення комплексної оцінки відгуків Інтернет-користувачів.

Проведено аналіз існуючих аналогів, на основі якого створено висновок про доцільність створення програмної системи.

Внаслідок проведеного аналізу наявних аналогів були сформульовані вимоги до розроблюваної системи, а саме функціональні вимоги, вимоги до користувацького інтерфейсу та використовуваних технологій.

Проведений аналіз мов та технологій програмування виявив обґрунтовану доцільність використання мови програмування Python та бібліотеки Keras для реалізації алгоритмів збору та аналізу відгуків й мережі аналізу емоційного забарвлення тексту. Інтерфейс користувача створено у вигляді Telegram боту з використанням бібілотеки python-telegram-bot.

Розроблена система для комплексної оцінки відгуків користувачів у мережі Інтернет:

- збирає відгуки з декількох ресурсів, для яких створені модулі збору даних на основі різних бібліотек;
- дозволяє редагувати список посилань на ресурси, на основі якого збираються відгуки;
- аналізує емоційне забарвлення тексту за декількома емоціями;
- формує звіт про результати аналізу посилань;
- формує звіт на основі історії збору та аналізу відгуків.

Програмна система розроблена в повному обсязі з відповідністю до розроблених вимог. Тестування системи виконано в повному обсязі згідно наявної методики тестування.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. SemanticForce [Електронний ресурс]. — Режим доступу: https://www.semanticforce.net/
2. AppFollow [Електронний ресурс]. — Режим доступу: https://appfollow.io/
3. Simpoll [Електронний ресурс]. — Режим доступу: https://simpoll.ru/
4. JagaJam [Електронний ресурс]. — Режим доступу: https://jagajam.com/
5. IqBuzz [Електронний ресурс]. — Режим доступу: http://iqbuzz.pro/
6. C # (programming language) [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/C_Sharp_(programming_language)
7. JavaScript [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/JavaScript
8. Python (programming language) [Електронний ресурс]. — Режим доступу : https://en.wikipedia.org/wiki/Python_(programming_language)
9. Microsoft Visual Studio [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio
10. IntelliSense in Visual Studio [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/visualstudio/ide/using-intellisense?view=vs-2019
11. Visual Studio Code [Електронний ресурс]. — Режим доступу: https://code.visualstudio.com/
12. Микросервисы (Microservices) [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/249183/
13. Telegram Bot API [Електронний ресурс]. — Режим доступу: https://core.telegram.org/bots/api
14. pyTelegramBotAPI. A simple, but extensible Python implementation for the Telegram Bot API [Електронний ресурс]. — Режим доступу: https://github.com/eternnoir/pyTelegramBotAPI

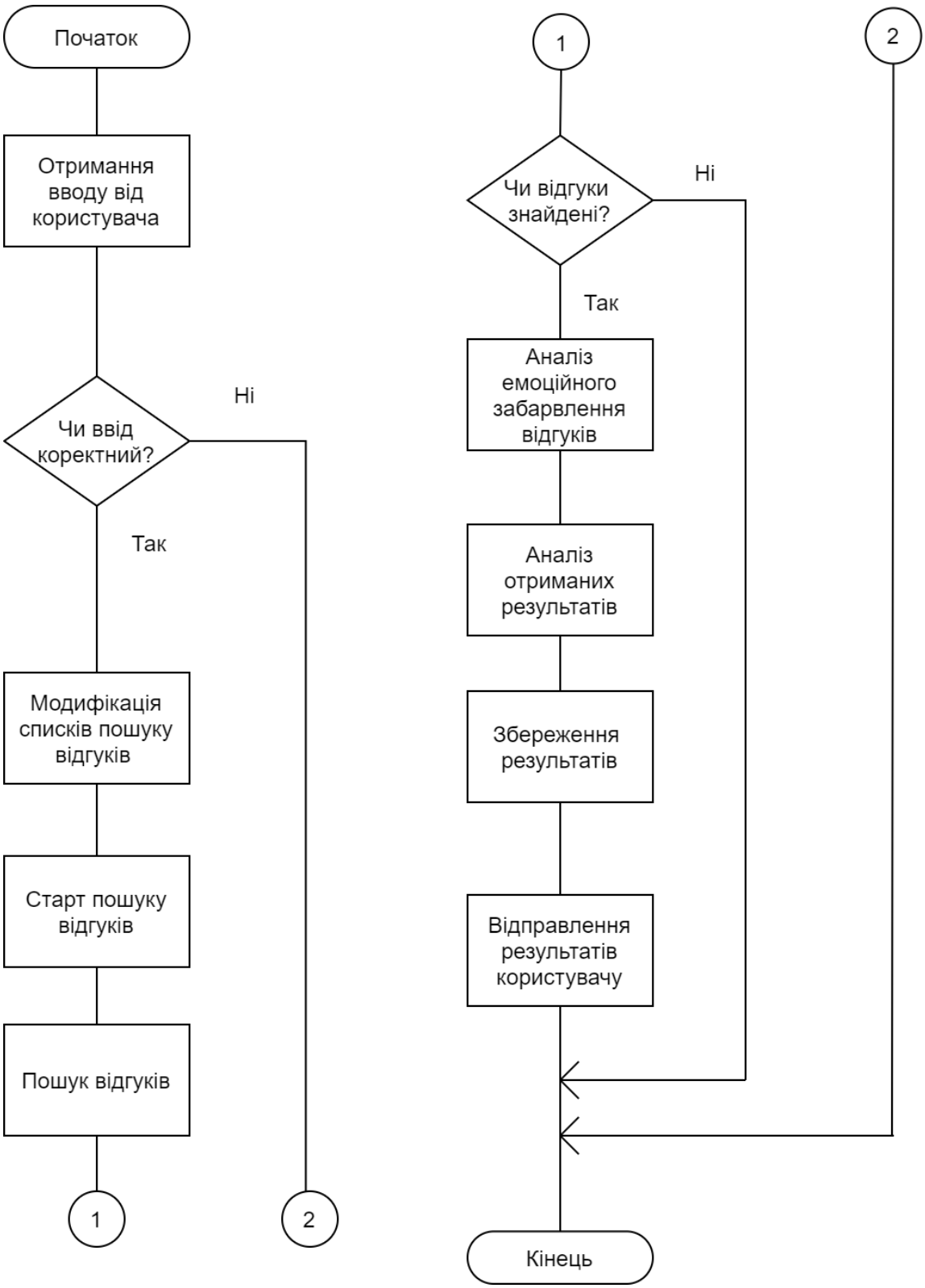
15. The PERMA Model: Your Scientific Theory of Happiness [Електронний ресурс]. — Режим доступу: https://positivepsychology.com/perma-model/
16. Keras [Електронний ресурс]. — Режим доступу: https://keras.io/
17. Параллельные вычисления CUDA [Електронний ресурс]. — Режим доступу: https://www.nvidia.com.ua/object/cuda-parallel-computing-ru.html
18. Stop words [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Stop_words
19. NLTK [Електронний ресурс]. — Режим доступу: https://www.nltk.org/
20. Lexical analysis. Tokenization [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Lexical_analysis#Tokenization
21. Stemming [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Stemming
22. Save and load Keras models [Електронний ресурс]. — Режим доступу: https://www.tensorflow.org/guide/keras/save_and_serialize
23. Colab [Електронний ресурс]. — Режим доступу: https://colab.research.google.com/
24. How to Save and Load Your Keras Deep Learning Model [Електронний ресурс]. — Режим доступу: https://machinelearningmastery.com/save-load-keras-deep-learning-models/
25. Web scraping [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Web_scraping
26. Dense layer [Електронний ресурс]. — Режим доступу: https://keras.io/api/layers/core_layers/dense/
27. Dropout — метод решения проблемы переобучения в нейронных сетях [Електронний ресурс]. — Режим доступу: https://habr.com/ru/company/wunderfund/blog/330814/
28. tf.keras.activations.sigmoid [Електронний ресурс]. — Режим доступу : https://www.tensorflow.org/api_docs/python/tf/keras/activations/sigmoid

29. BERT — state-of-the-art языковая модель для 104 языков. Туториал по запуску BERT локально и на Google Colab [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/436878/
30. Sentence Transformers: Multilingual Sentence Embeddings using BERT / RoBERTa / XLM-RoBERTa & Co. with PyTorch [Електронний ресурс]. — Режим доступу: https://github.com/UKPLab/sentence-transformers
31. World well being project [Електронний ресурс]. — Режим доступу: http://www.wwbp.org/lexica.html
32. Selectorlib [Електронний ресурс]. — Режим доступу: https://selectorlib.com/

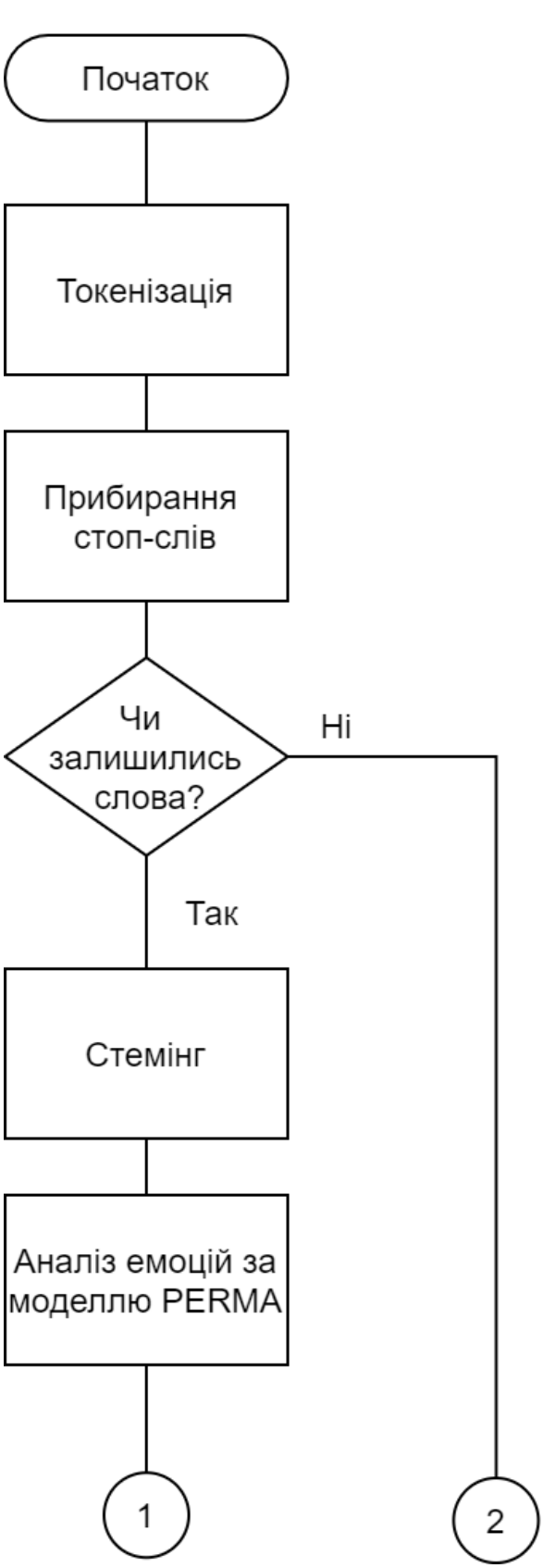
ДОДАТКИ

Додаток 1

Копії графічних матеріалів

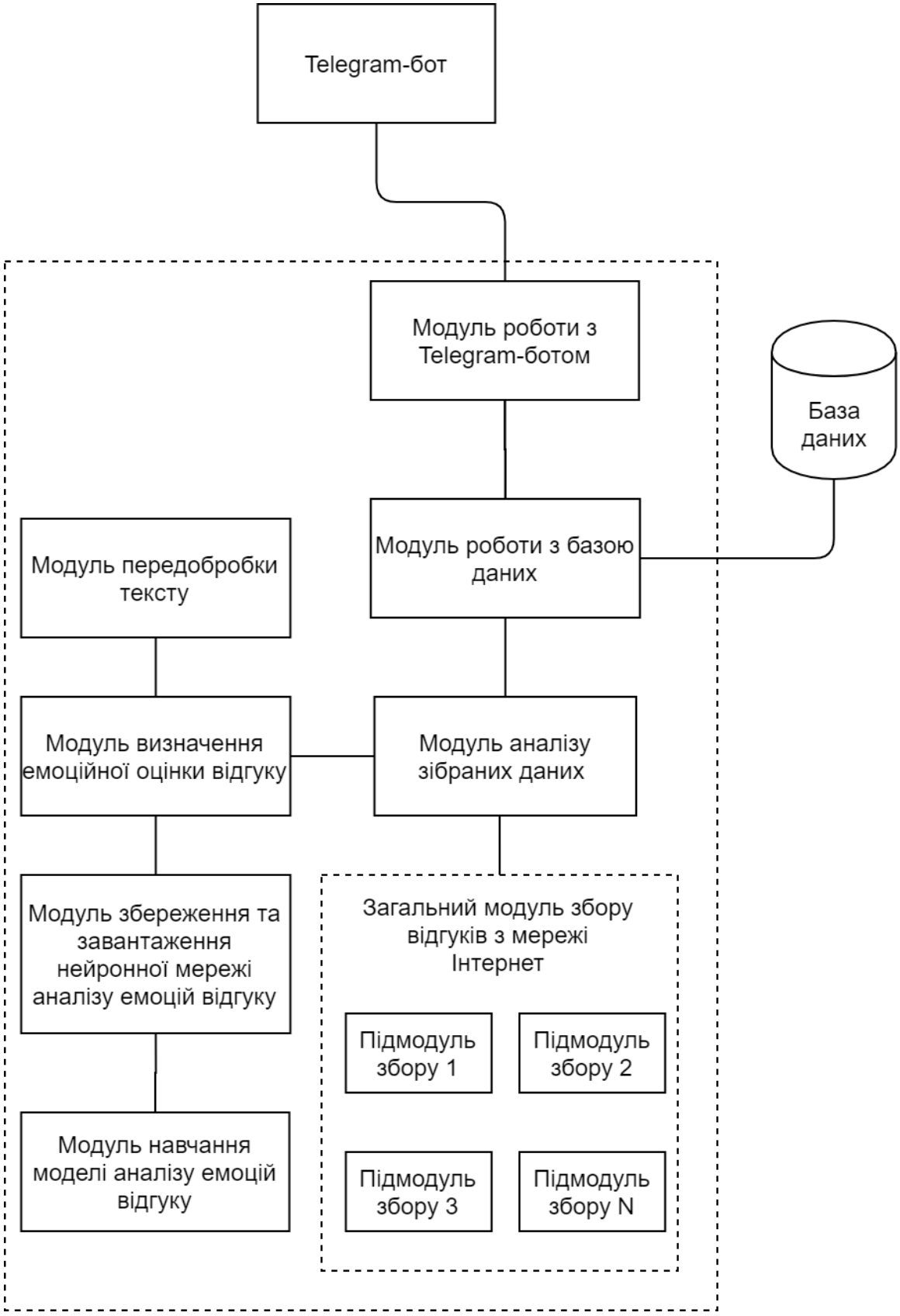


ДП.045440-06-99. Програмна система для визначення комплексної оцінки відгуків Інтернет-користувачів. Узагальнений алгоритм роботи системи. Схема алгоритму



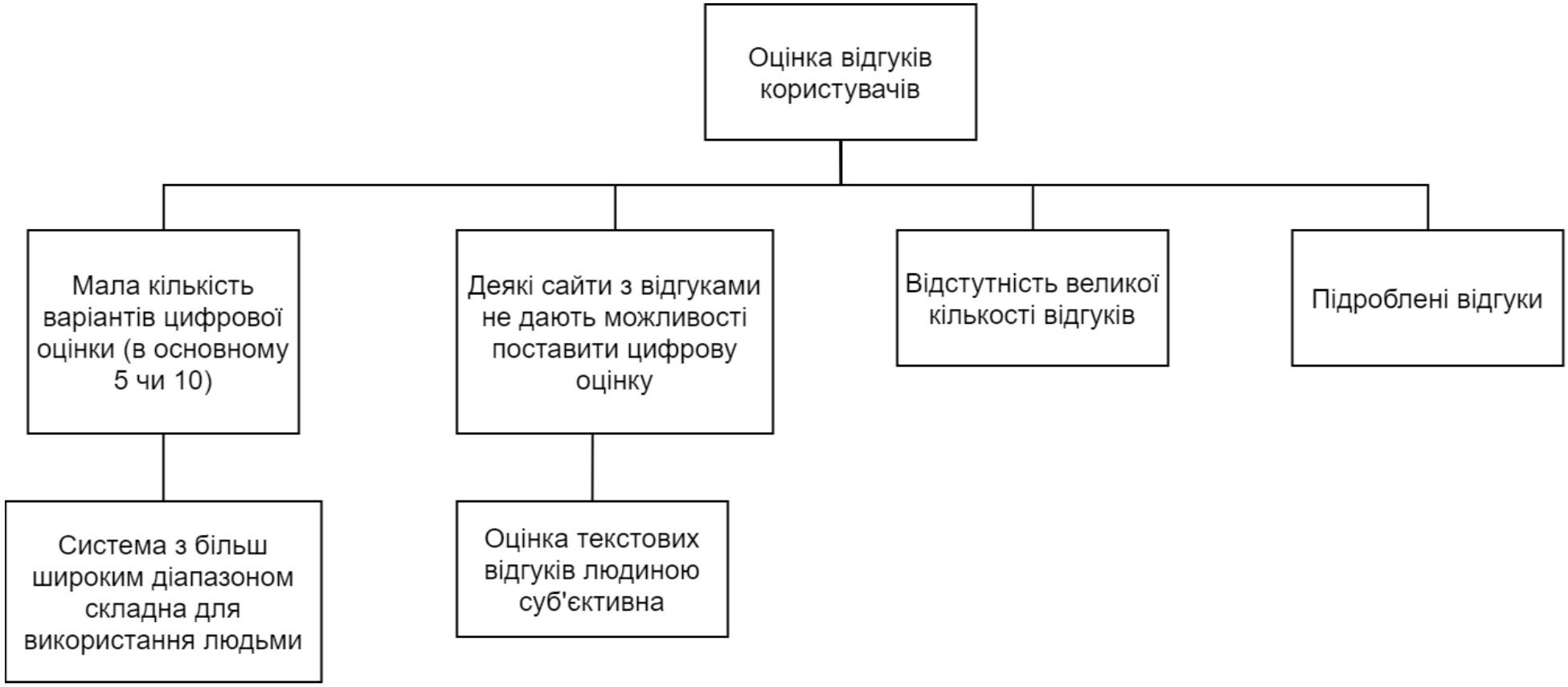
ДП.045440-07-99. Програмна система для визначення комплексної оцінки відгуків Інтернет-користувачів. Аналіз текстових відгуків. Схема алгоритму

ЗАГАЛЬНА СХЕМА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ВИЗНАЧЕННЯ КОМПЛЕКСНОЇ ОЦІНКИ ВІДГУКІВ ІНТЕРНЕТ-КОРИСТУВАЧІВ



Лук’янець М.О., група КП-62

ДЕРЕВО ПРОБЛЕМ



Лук’янець М.О., група КП-62

Додаток 2

Лістинг програми

```
import telebot
import time

from tld import exceptions
from tld import get_tld
from grand_parser import GrandParser

from mongo import Link, StatsTotal, StatsDomain
from mongoengine import connect

from matplot_test import get_plot, plot_history_one, plot_history_perma

import numpy as np

# in case of info logs change tmdb.py logging level SOMEHOW


connect('mydb')

parser = GrandParser()

bot = telebot.TeleBot("1017826671:AAGTLgZejY1Sm3mxlx8hKoa6TYbno0tIRJg")


def get_n_var(text: str, n=1):
    tokens = text.split()
    if n >= len(tokens):
        return None
    return tokens[n]


@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
    bot.send_message(message.chat.id, "Use commands for manipulating links
list and getting info")


@bot.message_handler(commands=['links'])
def get_links(m):
    reply = ""
    links = Link.objects
    if len(links) <= 0:
        reply = "Empty"
    else:
        for idx, link in enumerate(links):
            reply += f'{idx} - {link.title} \n'
    bot.send_message(m.chat.id, reply)


@bot.message_handler(commands=['add'])
def add_link(m):
    link = get_n_var(m.text)
    if not link:
        bot.send_message(m.chat.id, "No link")
        return

    try:
        supported = parser.check_parse_link(link)
    except exceptions.TldBadUrl:
        bot.send_message(m.chat.id, "Not valid link")
        return

    if not supported:
        bot.send_message(m.chat.id, "Not supported source")
```

```
        return

    link1 = Link(title=link)
    link1.save()
    bot.send_message(m.chat.id, "Link saved successfully")


@bot.message_handler(commands=['remove'])
def remove_link(m):
    try:
        index = int(get_n_var(m.text))
    except TypeError:
        bot.send_message(m.chat.id, "Not valid index")
        return

    links = Link.objects
    if len(links) <= index:
        bot.send_message(m.chat.id, "Not valid index")
        return

    links[index].delete()
    bot.send_message(m.chat.id, "Link deleted successfully")


@bot.message_handler(commands=['hist'])
def hist(m):
    if len(StatsTotal.objects) < 1:
        bot.send_message(m.chat.id, "No history")
        return
    bot.send_message(m.chat.id, "Started analyzing...")

    stats = StatsTotal.objects()

    ratings = [x.rating for x in stats if x.rating]

    vader = [x.vader for x in stats if x.vader]

    perma = [np.array(x.perma) for x in stats if x.perma]

    if len(ratings) != 0:
        plot_ratings = plot_history_one(ratings)
        plot_ratings.title('History of ratings', pad=-0.7)
        plot_ratings.savefig('temp.png')
        bot.send_photo(m.chat.id, open('temp.png', 'rb'))

    if len(vader) != 0:
        plot_vader = plot_history_one(vader)
        plot_vader.title('History of pos/neg emotions', pad=-0.7)
        plot_vader.savefig('temp.png')
        bot.send_photo(m.chat.id, open('temp.png', 'rb'))

    if len(perma) != 0:
        plot_perma = plot_history_perma(perma)
        plot_perma.title('History of PERMA', pad=-0.7)
        plot_perma.savefig('temp.png')
        bot.send_photo(m.chat.id, open('temp.png', 'rb'))


@bot.message_handler(commands=['analyze'])
def analyze(m):
    if len(Link.objects) < 1:
        bot.send_message(m.chat.id, "No links")
        return
```

```
    bot.send_message(m.chat.id, "Started analyzing...")

    stats = StatsDomain.objects(last=True)
    if len(stats) == 0:
        bot.send_message(m.chat.id, "No stats")
        return

    for stat in stats:
        response = ""
        domain = get_tld(stat.link, as_object=True).fld
        response += f'Result for {domain}: \n'
        if stat.rating:
            response += f'    Number marks(from 1 to 10):
{round(stat.rating, 2)}  \n'
        if stat.vader:
            response += f'    Emotions by negative or positive :
{round(stat.vader, 2)}  \n'
        bot.send_message(m.chat.id, response)
        if stat.perma:
            plt = get_plot(stat.perma)
            plt.title(domain + ' PERMA results')
            plt.savefig('temp.png')
            bot.send_photo(m.chat.id, open('temp.png', 'rb'))

    response = f'\nTotal results: \n'
    total_stats = StatsTotal.objects
    if len(total_stats) != 0:
        total = total_stats.order_by('-id').first()
        if total.rating:
            response += f'    Total number marks(from 1 to 10):
{round(total.rating, 2)}\n'
        if total.vader:
            response += f'    Total emotions by negative or positive :
{round(total.vader, 2)}  \n'
        bot.send_message(m.chat.id, response)
        if total.perma:
            plt = get_plot(total.perma)
            plt.title('Total PERMA results')
            plt.savefig('temp.png')
            bot.send_photo(m.chat.id, open('temp.png', 'rb'))


# @bot.message_handler(commands=['t'])
# def echo_all(m):
#     bot.send_message(m.chat.id, m.text)
#     time.sleep(10)
#     bot.reply_to(m, "second reply")


bot.polling()
import numpy as np
import matplotlib.pyplot as plt


def get_plot(arr):
    n = 5
    # arr = [0.00965389, 0.01049378, 0.00429987, 0.01548897, 0.01036124,
0.01478354, 0.01471009, 0.01199437, 0.02329957,
    #        0.02300449]
    start = np.array(arr) * 40

    X = np.arange(n)
    pos = start[5: 10]
    neg = start[0: 5]
```

```
    fig, ax = plt.subplots()
    ax.bar(['A', 'E', 'M', 'P', 'R'], +pos, facecolor='#9999ff',
edgecolor='white')
    ax.bar(['A', 'E', 'M', 'P', 'R'], -neg, facecolor='#ff9999',
edgecolor='white')

    for x, y in zip(X, pos):
        ax.text(x, y + 0.05, '%.2f' % y, ha='center', va='bottom')

    for x, y in zip(X, neg):
        ax.text(x, -y - 0.15, '%.2f' % y, ha='center', va='bottom')

    ax.set_ylim(-1.25, +1.25)

    return plt


def plot_history_one(stats):
    t = np.arange(0, len(stats))
    line = np.array(stats)

    fig, axs = plt.subplots()
    axs.plot(t, line)
    axs.set_xlabel('times')
    axs.set_ylabel('results')
    axs.grid(True)
    fig.tight_layout()
    return plt


def plot_history_perma(stats):
    t = np.arange(0, len(stats))

    arr = np.array(stats) * 40
    NEG_A = -arr[:, 0]
    NEG_E = -arr[:, 1]
    NEG_M = -arr[:, 2]
    NEG_P = -arr[:, 3]
    NEG_R = -arr[:, 4]
    POS_A = arr[:, 5]
    POS_E = arr[:, 6]
    POS_M = arr[:, 7]
    POS_P = arr[:, 8]
    POS_R = arr[:, 9]

    fig, axs = plt.subplots()
    axs.plot(t, NEG_A, label='NEG_A')
    axs.plot(t, NEG_E, label='NEG_E')
    axs.plot(t, NEG_M, label='NEG_M')
    axs.plot(t, NEG_P, label='NEG_P')
    axs.plot(t, NEG_R, label='NEG_R')
    axs.plot(t, POS_A, label='POS_A')
    axs.plot(t, POS_E, label='POS_E')
    axs.plot(t, POS_M, label='POS_M')
    axs.plot(t, POS_P, label='POS_P')
    axs.plot(t, POS_R, label='POS_R')

    axs.set_xlabel('times')
    axs.set_ylabel('perma marks')
    axs.grid(True)
    axs.legend()

    fig.tight_layout()
```

```
    return plt

import pandas as pd
from sentence_transformers import SentenceTransformer
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout

import net_save_load

df = pd.read_csv('perma_new_noALPHA.csv')
df.head()

model = SentenceTransformer('bert-base-nli-mean-tokens')

# df = df.sample(frac=1)

Y = df.drop(columns=['term']).to_numpy()
encoded = model.encode(df.term)
encoded = np.array(encoded)

# x_val = encoded[-500:]
# y_val = Y[-500:]
# encoded = encoded[:-500]
# Y = Y[:-500]


net = Sequential([Dense(256, input_shape=(768,)), Activation('sigmoid'),
Dropout(0.2),
                  Dense(64), Activation('sigmoid'), Dropout(0.2),
                  Dense(10)])

net.compile(optimizer='rmsprop',
            loss='mse',
            metrics=['accuracy'])

for layer in net.layers:
    print(layer.output_shape)

hist = net.fit(encoded, Y, epochs=10, verbose=2)  # validation_data=(x_val,
y_val)

scores = net.evaluate(encoded, Y, verbose=0)
print("%s: %.2f%%" % (net.metrics_names[1], scores[1] * 100))

enc = model.encode(["what a famous"])
enc = enc[0]
net.predict([[enc]])

net_save_load.save_net(net)
from omdbapi.movie_search import GetMovie
from basic_parser import BasicParser
import pandas as pd


class OmdbScraper(BasicParser):
    domain = "omdbapi.com"
    max_rating = 10
    api_key = '5cbd8e01'

    def parse(self, link: str):
        if not self.check_parse_link(link):
            return False
        movie = GetMovie(title=link.split('/')[-1], api_key=self.api_key)
```

```
        # print(movie)
        # print('SPLIT')
        # print(movie.get_all_data())
        ratings = []
        data = movie.get_all_data()
        imdb = data['imdbRating']
        meta = data['Metascore']
        tomato = data['Ratings'][1]['Value'].replace('%', '')
        ratings.append(imdb)
        ratings.append(int(meta)/10)
        ratings.append(int(tomato)/10)

        movie_df = pd.DataFrame({'ratings': ratings})
        return movie_df

import requests
from bs4 import BeautifulSoup

import time
import random as rand

import pandas as pd

from basic_parser import BasicParser


class MetacriticScraper(BasicParser):
    domain = "metacritic.com"
    max_rating = 10

    def parse(self, link: str):
        if not self.check_parse_link(link):
            return False
        review_dict = {'ratings': [], 'reviews': []}

        user_agent = {'User-agent': 'Mozilla/5.0'}
        response = requests.get(link, headers=user_agent, timeout=10)
        soup_page = BeautifulSoup(response.text, 'html.parser')
        page = 1
        if soup_page.find('li', class_='last_page') is not None:
            page = soup_page.find('li', class_='last_page').find('a').text

        for page in range(0, int(page)):
            url = link + '?page=' + str(page)
            user_agent = {'User-agent': 'Mozilla/5.0'}
            response = requests.get(url, headers=user_agent, timeout=10)
            # time.sleep(rand.randint(3,30))
            soup = BeautifulSoup(response.text, 'html.parser')
            for review in soup.find_all('div', class_='review pad_top1'):
                if review.find('div', class_='left fl') is None:
                    break
                review_dict['ratings'].append(review.find('div',
class_='left fl').find_all('div')[0].text)
                if review.find('span', class_='blurb blurb_expanded'):
                    review_dict['reviews'].append(review.find('span',
class_='blurb blurb_expanded').text)
                else:
                    review_dict['reviews'].append(review.find('div',
class_='review_body').find('span').text)

        movie_df = pd.DataFrame(review_dict)
        return movie_df

from mongoengine import *
```

```
connect('mydb')


class Link(Document):
    title = StringField(required=True, max_length=200)


class StatsDomain(Document):
    link = StringField(required=True, max_length=200)
    rating = FloatField()
    perma = ListField()
    vader = FloatField()
    last = BooleanField(required=True)


class StatsTotal(Document):
    rating = FloatField()
    perma = ListField(FloatField())
    vader = FloatField()
from basic_parser import BasicParser
from selectorlib import Extractor
import requests
import pandas as pd


class AmazonScraper(BasicParser):
    domain = "amazon.com"
    max_rating = 10

    e = Extractor.from_yaml_file('selectors.yml')

    def __scrape__(self, url):
        headers = {
            'authority': 'www.amazon.com',
            'pragma': 'no-cache',
            'cache-control': 'no-cache',
            'dnt': '1',
            'upgrade-insecure-requests': '1',
            'user-agent': 'Mozilla/5.0',
            'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apn
g,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
            'sec-fetch-site': 'none',
            'sec-fetch-mode': 'navigate',
            'sec-fetch-dest': 'document',
            'accept-language': 'en-GB,en-US;q=0.9,en;q=0.8',
        }

        # Download the page using requests
        # print("Downloading %s" % url)
        r = requests.get(url, headers=headers)
        # print(r.text)
        if "To discuss automated access to Amazon data please contact" in
r.text:
            print("Page %s was blocked by Amazon. Please try using better
proxies\n" % url)
            return None
        # Simple check to check if page was blocked (Usually 503)
        if r.status_code > 500:
            if "To discuss automated access to Amazon data please contact"
in r.text:
                print("Page %s was blocked by Amazon. Please try using
better proxies\n" % url)
```

```
            else:
                print("Page %s must have been blocked by Amazon as the
status code was %d" % (url, r.status_code))
            return None
        # Pass the HTML of the page and create
        return self.e.extract(r.text)

    def parse(self, link: str):
        if not self.check_parse_link(link):
            return False
        data = self.__scrape__(link)
        ratings = []
        reviews = []
        if data:
            for r in data['reviews']:
                rating = r['rating'].split(' out of')[0]
                rating = float(rating) * 2
                ratings.append(float(rating))
                reviews.append(r['content'])
        movie_df = pd.DataFrame({'ratings': ratings,
                                 'reviews': reviews,})
        return movie_df
import numpy as np
import pandas as pd

df = pd.read_csv("datasets/perma_new_noALPHA.csv", encoding='utf-8')

vars = [' NEG_A', ' NEG_E', ' NEG_M', ' NEG_P', ' NEG_R',
        ' POS_A', ' POS_E', ' POS_M', ' POS_P', ' POS_R ']

for var in vars:
    print(var + ": " + str(df[var].abs().sum()))

df = pd.read_csv("perma_test.csv", encoding='utf-8')

for index, row in df.iterrows():
    print(row)

from sentence_transformers import SentenceTransformer

model = SentenceTransformer('bert-base-nli-mean-tokens')

df = pd.read_csv("perma_new.csv", encoding='utf-8')

Y = df.iloc[1:].to_numpy()

encoded = model.encode(df.term)
print(encoded)
encoded = np.array(encoded)

print(encoded)


df = pd.read_csv("perma_semicol.csv", encoding='utf-8')

dict_words = {}
for index, row in df.iterrows():
    if row['term'] not in dict_words:
        dict_words[row['term']] = {'NEG_A': 0, 'NEG_E': 0, 'NEG_M': 0,
'NEG_P': 0, 'NEG_R': 0,
                                   'POS_A': 0, 'POS_E': 0, 'POS_M': 0,
'POS_P': 0, 'POS_R': 0}
    dict_words[row['term']][row['category']] = row['weight']
```

```
f = open("perma_new_noALPHA.csv", 'a', encoding='utf-8')

f.write('term, NEG_A, NEG_E, NEG_M, NEG_P, NEG_R, POS_A, POS_E, POS_M,
POS_P, POS_R \n')

for key in dict_words.keys():
    if key.replace(' ', '').isalpha() and not any(ext in key for ext in
['NEG_A', 'NEG_E', 'NEG_M', 'NEG_P', 'NEG_R',

'POS_A', 'POS_E', 'POS_M', 'POS_P', 'POS_R']):
        val = list(dict_words[key].values())
        f.write(f'{key}, {val[0]}, {val[1]}, {val[2]}, {val[3]}, {val[4]},
'
                f'{val[5]}, {val[6]}, {val[7]}, {val[8]}, {val[9]}\n')

f.close()

f = open('perma_new.csv')
Lines = f.readlines()

f = open("perma_new_semicol.csv", "a")

for line in Lines:
    line = line[:-2] + '\n'
    f.write(line)


f.close()


f = open('perma_comma.csv')
Lines = f.readlines()

f = open("perma_first_comma.csv", "a")

for line in Lines:
    if line.startswith(','):
        line = line[1:]
    f.write(line)


f.close()


f = open('perma_first_comma.csv')
Lines = f.readlines()

f = open("perma_minus_comma.csv", "a")

for line in Lines:
    if line.count(",") == 4:
        line = line.replace(",", "", 1)
    f.write(line)


f.close()

import emotions_analyze
import net_save_load
from sentence_transformers import SentenceTransformer
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from grand_parser import GrandParser

# TEST DB
```

```
from mongoengine import connect
from mongo import Link

connect('mydb')

# link1 = Link(title = "VERY TEST LINK")
#
# link1.save()

Link.objects(title="VERY TEST LINK").delete()

for link in Link.objects:
    print(link.title)

TEST IMDB

print(GrandParser().check_parse_link("http://www.imdb.com/0816692"))
print(GrandParser().parse("http://www.imdb.com/0816692"))

TEST OMDB

print(GrandParser().check_parse_link("www.http://www.omdbapi.com/Interstell
ar"))
print(GrandParser().parse("www.http://www.omdbapi.com/Interstellar"))


TEST METACRITIC

print(GrandParser().check_parse_link("https://www.metacritic.com/movie/scoo
b!/user-reviews"))
print(GrandParser().parse("https://www.metacritic.com/movie/scoob!/user-
reviews"))

TEST METACRITIC

print(GrandParser().check_parse_link("https://www.metacritic.com/movie/star
-wars-episode-iv---a-new-hope/user-reviews"))
print(GrandParser().parse("https://www.metacritic.com/movie/star-wars-
episode-iv---a-new-hope/user-reviews"))


TEST TMDB

print(GrandParser().check_parse_link("https://www.themoviedb.org/movie/3851
03-scooby-doo?language=en-US"))
print(GrandParser().parse("https://www.themoviedb.org/movie/385103-scooby-
doo?language=en-US"))


TEST AMAZON

print(GrandParser().check_parse_link("https://www.amazon.com/Sherlock-
Holmes-Robert-Downey-Jr/dp/B0038OY6DW/ref"

"=cm_cr_arp_d_bdcrb_top?ie=UTF8#customer-review-section"))
print(GrandParser().parse("https://www.amazon.com/Sherlock-Holmes-Robert-
Downey-Jr/dp/B0038OY6DW/ref"
                          "=cm_cr_arp_d_bdcrb_top?ie=UTF8#customer-review-
section"))

data = GrandParser().parse("https://www.amazon.com/Sherlock-Holmes-Robert-
Downey-Jr/dp/B0038OY6DW/ref"
```

```
                                "=cm_cr_arp_d_bdcrb_top?ie=UTF8#customer-review-
section")

net = net_save_load.load_model()
model = SentenceTransformer('bert-base-nli-mean-tokens')

for rev in data['reviews']:
    print(emotions_analyze.emotions_marks(rev, net, model))


TEST ANALYZERS

analyser = SentimentIntensityAnalyzer()

score = analyser.polarity_scores("Even the funniest movies eventually stop
making me laugh after I've watched them
enough times that the humor no longer surprises me. A joke never has the
same effect when you know the punch line
in advance. But every once in a blue moon, a comedy comes along that is so
thoughtful and meaningful in addition to
being funny that after seeing it a dozen times and laughing less often, I
start noticing its depth and insight. For
me, no movie has so perfectly united hilarity with profundity as Groundhog
Day, which happens to be my favorite
movie of all time.") print(score)

net = net_save_load.load_model()
model = SentenceTransformer('bert-base-nli-mean-tokens')

print(emotions_analyze.emotions_marks("Even the funniest movies eventually
stop making me laugh after I've watched
them enough times that the humor no longer surprises me. A joke never has
the same effect when you know the punch
line in advance. But every once in a blue moon, a comedy comes along that
is so thoughtful and meaningful in
addition to being funny that after seeing it a dozen times and laughing
less often, I start noticing its depth and
insight. For me, no movie has so perfectly united hilarity with profundity
as Groundhog Day, which happens to be my
favorite movie of all time.", net, model))
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ КОМПЛЕКСНОЇ ОЦІНКИ ВІДГУКІВ ІНТЕРНЕТ-КОРИСТУВАЧІВ

Виконав: Лук’янець Михайло Олександрович

Науковий керівник: доцент кафедри ПЗКС, к.т.н., доцент, Заболотня Т.М.

Київ – 2020

МЕТА

Мета проєкту: розробити програмну систему для визначення комплексної оцінки відгуків Інтернет-користувачів

АКТУАЛЬНІСТЬ

- У багатьох користувачів з’явилася звичка як залишати відгуки про товари, так і шукати відгуки перед вибором товару
- Існує багато ресурсів, на яких користувачі можуть залишати відгуки
- Кількість таких ресурсів постійно збільшується, що ускладнює аналіз відгуків
- Система відгуків на різних ресурсах має свою специфіку, що також ускладнює аналіз відгуків

АНАЛОГИ

SemanticForce
моніторинг та аналіз
Інтернет-медіа

AppFollow
моніторинг за додатками
для телефонів

Simpoll
сервіс створення форм для
збору відгуків та їх аналізу

IQ Buzz
моніторинг згадок про компанію
у ЗМІ та соцмережах

ПОСТАНОВКА ЗАДАЧІ

Завдання:

- Вибрати спосіб та засоби реалізації та створити модуль аналізу текстових відгуків
- Обрати СКБД та створити модуль зв’язку з БД
- Знайти декілька джерел відгуків та розробити способи отримання даних з цих джерел
- Обрати інтерфейс користувача та реалізувати його
- Створити методику тестування та протестувати систему згідно методики

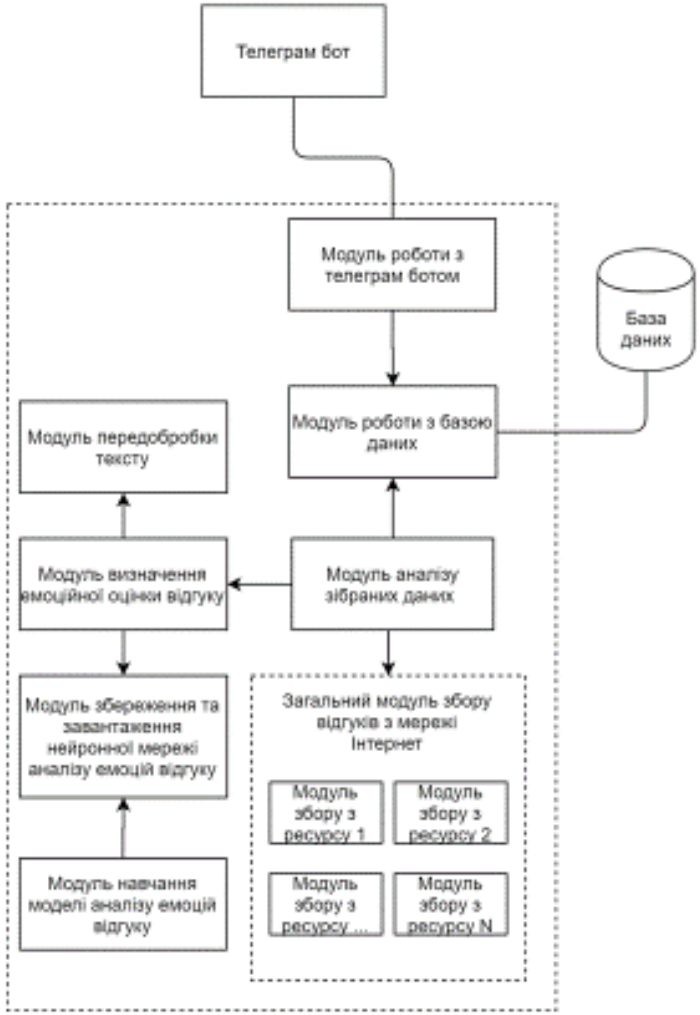
ДЕРЕВО ПРОБЛЕМ

ЗАСОБИ РЕАЛІЗАЦІЇ

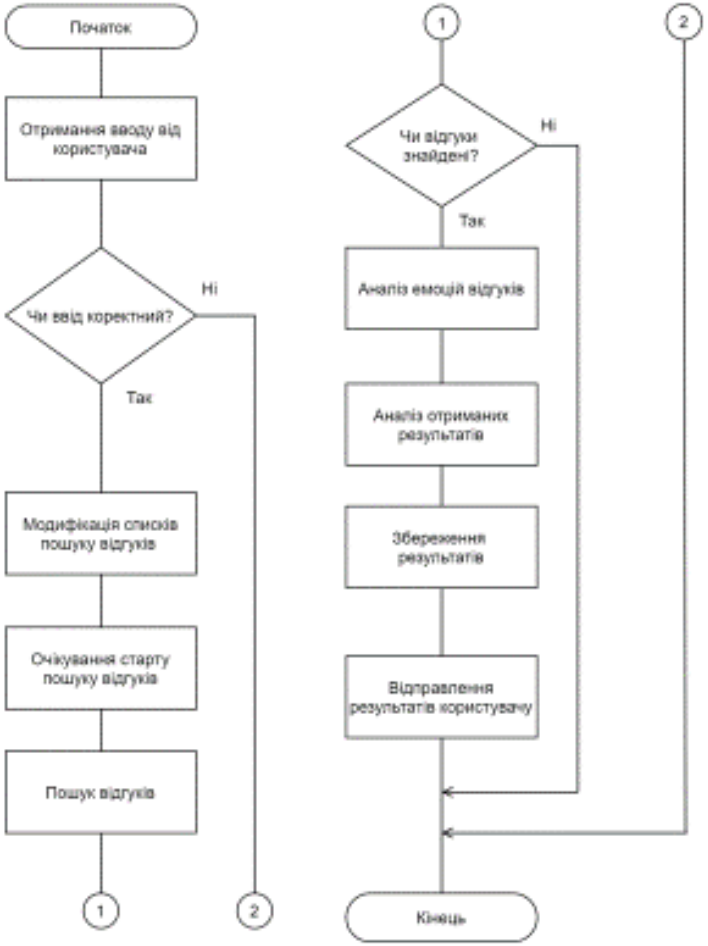
NLTK

mongoengine

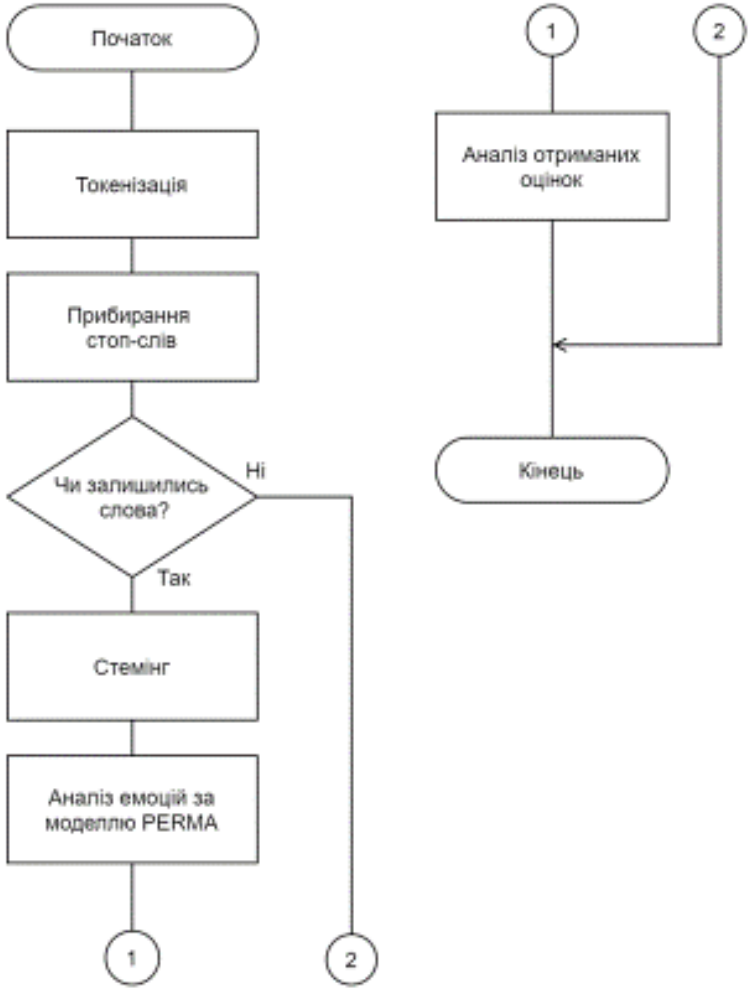
СТРУКТУРА СИСТЕМИ



АЛГОРИТМ РОБОТИ СИСТЕМИ

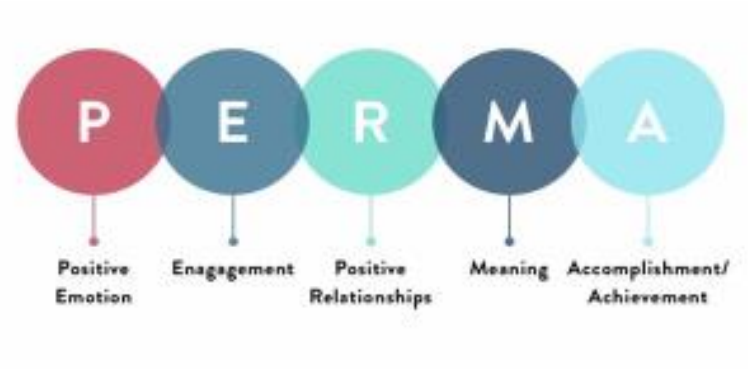


АЛГОРИТМ АНАЛІЗУ ТЕКСТОВИХ ВІДГУКІВ



ДЖЕРЕЛА, З ЯКИХ ЗБИРАЛИСЬ ВІДГУКИ

НАБІР ЕМОЦІЙ, ЗА ЯКИМ ПРОВОДИТЬСЯ АНАЛІЗ

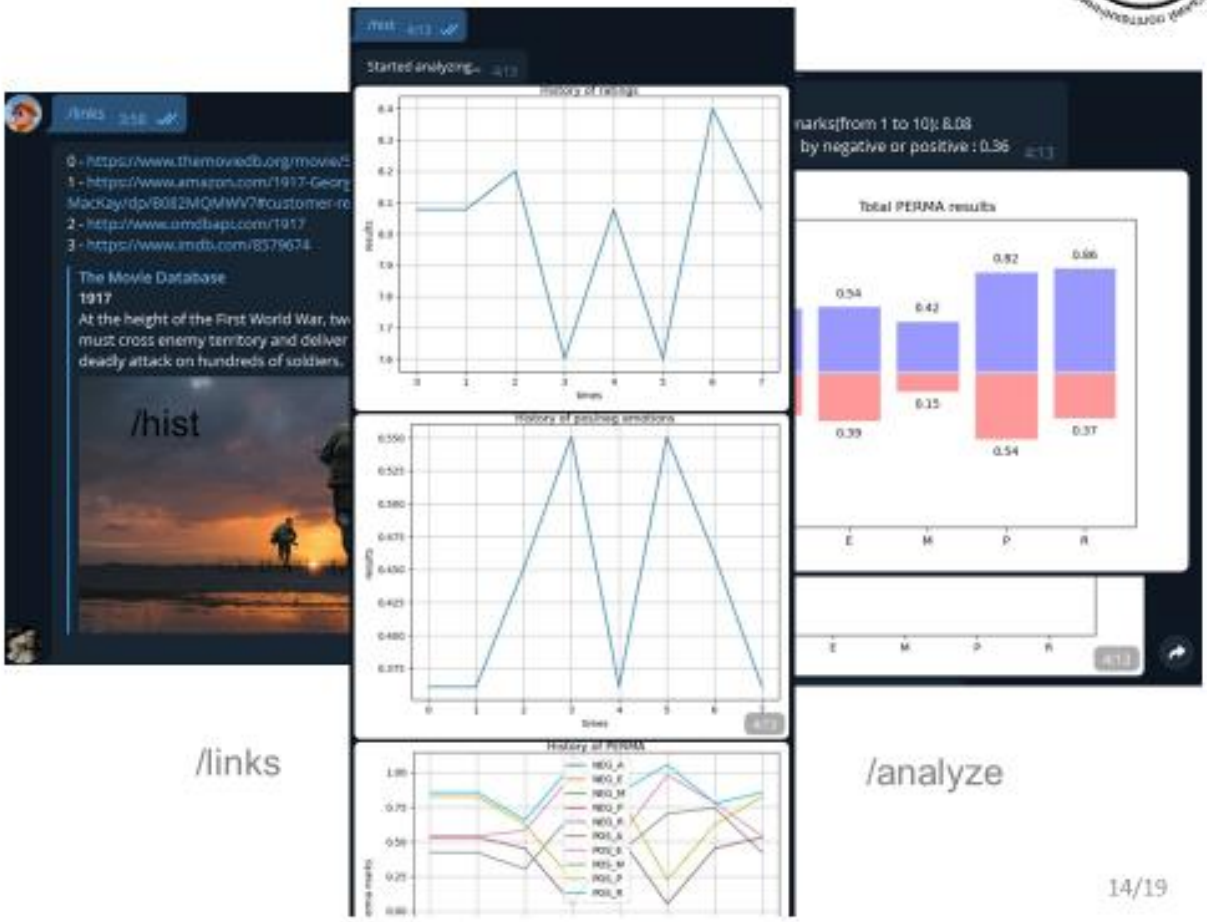


	term	NEG_A	NEG_E	NEG_M	NEG_P	NEG_R	POS_A	POS_E	POS_M	POS_P	POS_R
0	to throw	0.035770	0.062716	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000
1	my house	0.000000	0.081765	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000
2	sleep	0.046107	0.220759	0.071719	0.080871	0.000000	0.0	0.000000	0.000000	0.000000	0.000000
3	to blow	0.000000	0.039024	0.029119	0.049670	0.058226	0.0	0.000000	0.000000	0.000000	0.000000
4	and i	-0.035114	-0.036638	0.000000	0.016437	-0.070688	0.0	-0.038926	-0.028717	-0.030241	0.075878

ПРИКЛАД АНАЛІЗУ ВІДГУКІВ

ІНТЕРФЕЙС

/add →



/links

/analyze

АПРОБАЦІЯ

INTELLECTUAL SYSTEMS OF DECISION-MAKING AND PROBLEMS OF COMPUTATIONAL INTELLIGENCE (**ISDMCI**'2020)

ВИСНОВКИ

- Проаналізовано аналоги
- Проаналізовано та обрано засоби розроблення
- Визначено вимоги до розроблюваної системи
- Розроблено структуру системи
- Створено модуль аналізу текстових відгуків за багатьма емоціями

ВИСНОВКИ

- Показано проблеми та особливості, які виникли при розробленні системи
- Розроблено модулі збору даних для п’яти джерел відгуків
- Створено інтерфейс для взаємодії з користувачем у вигляді телеграм боту
- Протестовано розроблену систему згідно програмі та методиці тестування
- Запропоновано напрямки подальших досліджень

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2019 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ КОМПЛЕКСНОЇ ОЦІНКИ ВІДГУКІВ ІНТЕРНЕТ-КОРИСТУВАЧІВ

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Михайло ЛУК'ЯНЕЦЬ

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмна система для визначення комплексної оцінки відгуків Інтернет-користувачів., яка являє собою сервер та інтерфейс у вигляді телеграм бота, створену на платформі Python з використанням технології NLTK та Keras.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) зв’язок з базою даних, у якій зберігається список посилань та історія аналізів;
2) працездатність інтерфейсу для редагування списку посилань;
3) аналіз текстових відгуків за системою PERMA;
4) забезпечення збору відгуків з ресурсів, для яких реалізовано модулі збору;
5) зручність роботи з телеграм ботом;
6) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію PyCharm.

Працездатність програмної системи перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування інтерфейсу телеграм боту виконанням команд;
5) тестування параллельних запитів до телеграм боту;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2020 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ВИЗНАЧЕННЯ КОМПЛЕКСНОЇ ОЦІНКИ ВІДГУКІВ ІНТЕРНЕТ-КОРИСТУВАЧІВ

Керівництво користувача

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Тетяна ЗАБОЛОТНЯ

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

____ Михайло ЛУК'ЯНЕЦЬ

2020

ЗМІСТ

1. Опис структури телеграм боту

Інтерфейс користувача для програмної системи аналізу відгуків Інтернет- користувачів складається із телеграм боту. Мова інтерфейсу користувача англійська. Телеграм бот генерує відповідь в залежності від введеної команди та додаткової інформації для команди.

2. Опис вмісту телеграм боту

Інтерфейс телеграм боту (рис. 1) складається з кнопки для показу доступних команд з їх описом для використання та поля для вводу команд .

До списку команд входять такі команди:

- «help»;
- «links»;
- «add»;
- «analyze»;
- «hist»
- «remove».

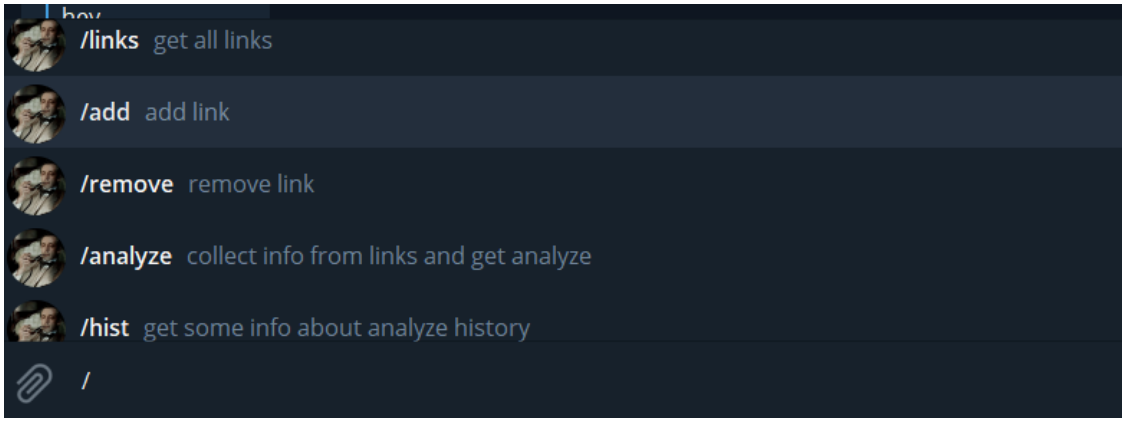


Рис. 1. Інтерфейс телеграм боту

3. Процедура редагування списку посилань

Редагування списку посилань виконується за допомогою зазначених команд «add», «remove», «links».

Команда «links» не потребує додаткових параметрів для використання. Команда виводить список посилань, що на даний час знаходяться у списку для пошуку відгуків (рис. 2).

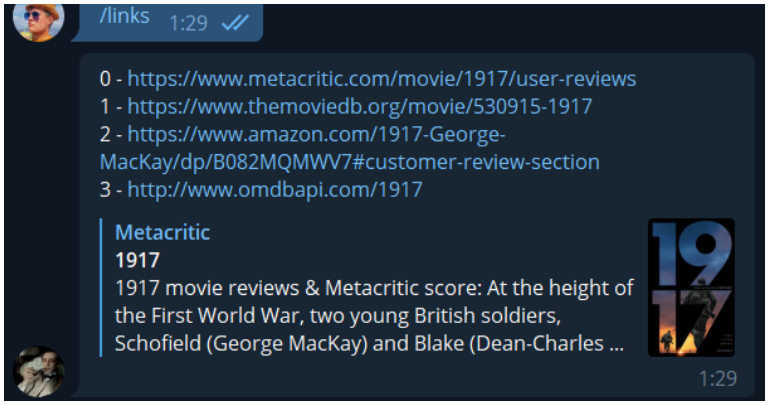


Рис. 2. Результат виконання команди «links»

Команда «add» потребує посилання для додавання до списку посилань для збору та аналізу відгуків. Посилання має бути на один із ресурсів, збір відгуків з якого підтримує система. У разі невідповідності посилання або його відсутності користувач отримає різні повідомлення про помилку. У разі нормальної роботи користувач отримає повідомлення про те, що посилання було успішно додане (рис. 3).

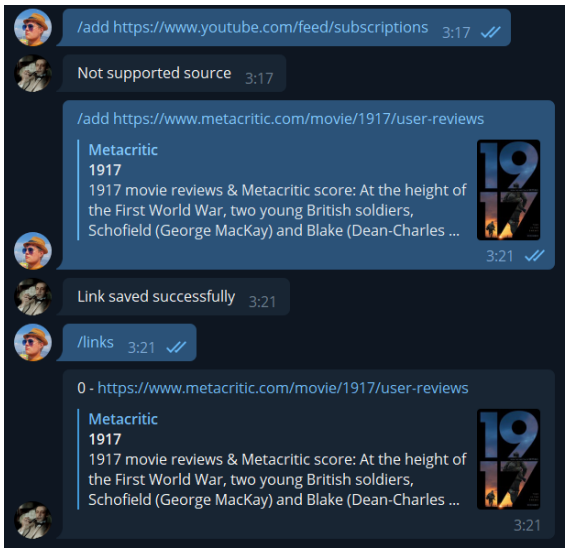


Рис. 3. Результат виконання команди «add»

Команда «remove» потребує номер посилання для видалення зі списку посилань для збору та аналізу відгуків. Номер має відповідати номеру у списку посилань. У разі невідповідності номеру або його відсутності користувач отримає різні повідомлення про помилку. У разі нормальної роботи користувач отримає повідомлення про те, що посилання було успішно видалене (рис. 4).

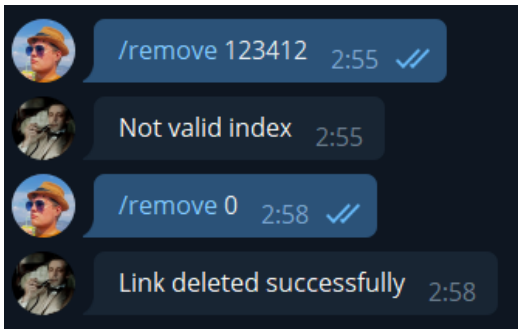


Рис. 4. Результат виконання команди «remove»

4. Отримання результатів аналізу відгуків за посиланнями

Команда «analyze» не потребує додаткових параметрів для використання. Команда виводить результати аналізу відгуків до посилань, що на даний час знаходяться у списку для пошуку відгуків (рис. 5). Результати виводяться для кожного посилання окремо та разом (рис. 6).

Виведення результатів залежить від наявності цифрових та текстових відгуків. У разі відсутності одного або іншого буде відсутній аналіз цифрових або емоційного забарвлення текстових відгуків відповідно.

Аналіз емоційного забарвлення емоцій за системою PERMA виводиться у вигляді графіку, на якому зверху синіми стовпцями позначені позитивні оцінки за кожною з емоцій, а знизу червоними стовпцями позначені негативні оцінки за кожною з емоцій з системи PERMA (рис. 7).

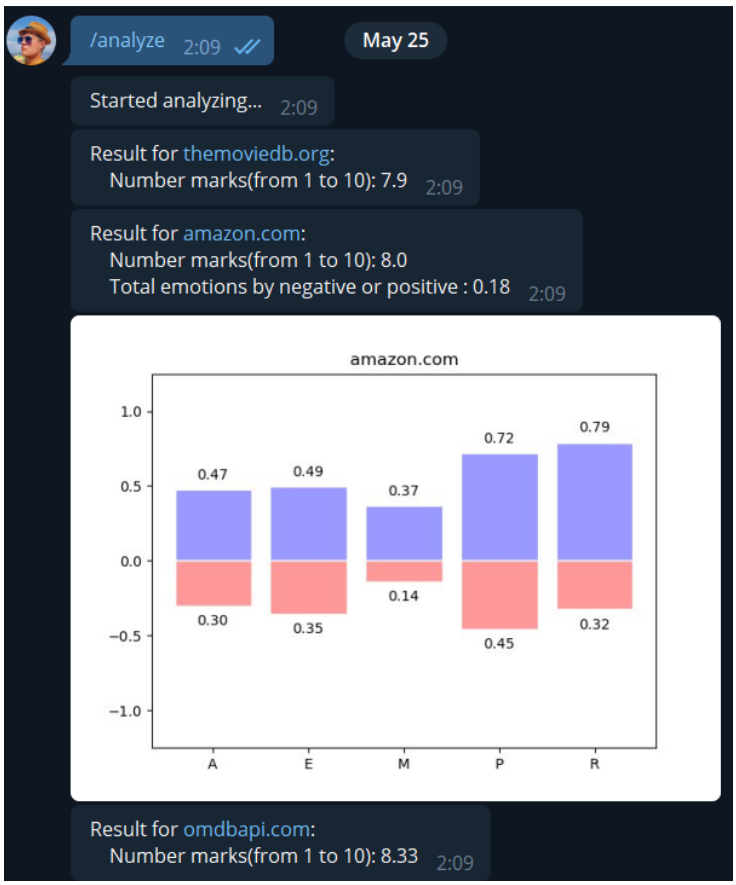


Рис. 5. Результат виконання команди «analyze»

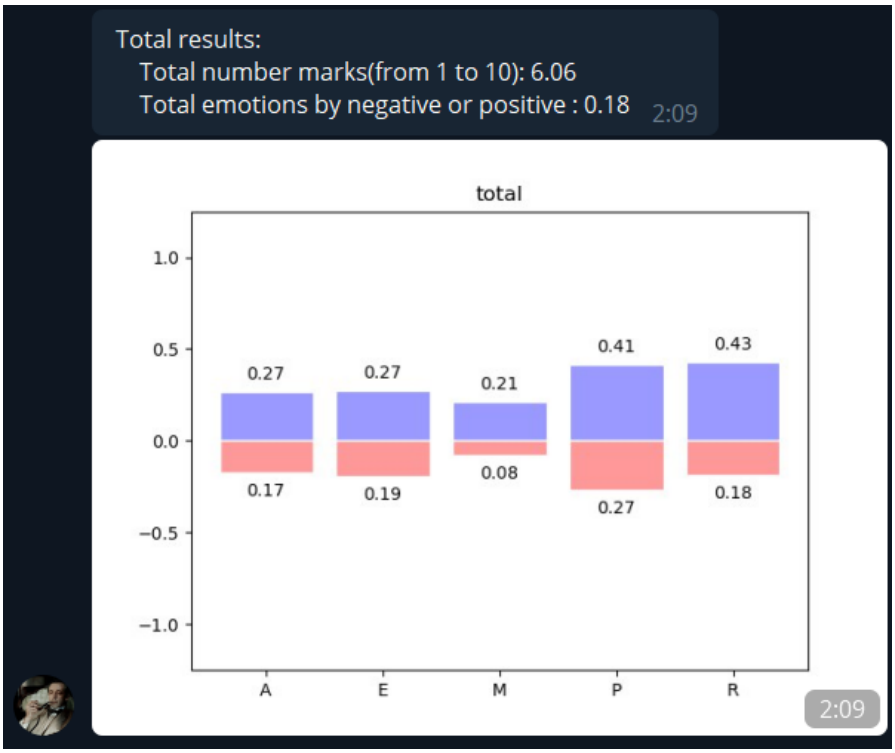


Рис. 6. Результат виконання команди «analyze» для загальної оцінки

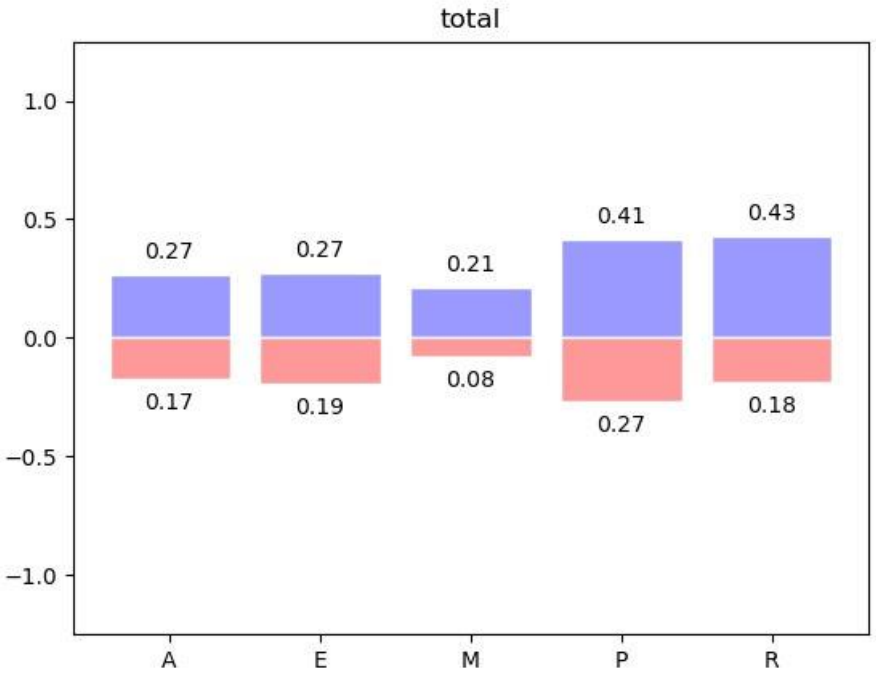


Рис. 7. Таблиця аналізу відгуків за системою PERMA

5. Отримання звіту по історії аналізу посилань

Команда «hist» не потребує додаткових параметрів для використання. Команда виводить результати історії аналізів відгуків до посилань, а саме історії загальних результатів (рис. 5).

У результаті виконання виведуться три графіки за наявності цифрових та емоційних оцінок у історії. Перший графік демонструє історію зміни цифрової оцінки користувачів (рис. 8). Другий графік виводить історію зміни емоційної оцінки відгуків системою Vader від NLTK (рис. 9). Третій графік показує історію зміни оцінок системи PERMA за десятьма параметрами. Зверху лініями позначені позитивні оцінки за кожною з емоцій, а знизу іншими лініями позначені негативні оцінки за кожною з емоцій з системи PERMA (рис. 10). Для кожного з параметрів для виводу є відповідне позначення відповідності кольору та параметру на легенді графіку.

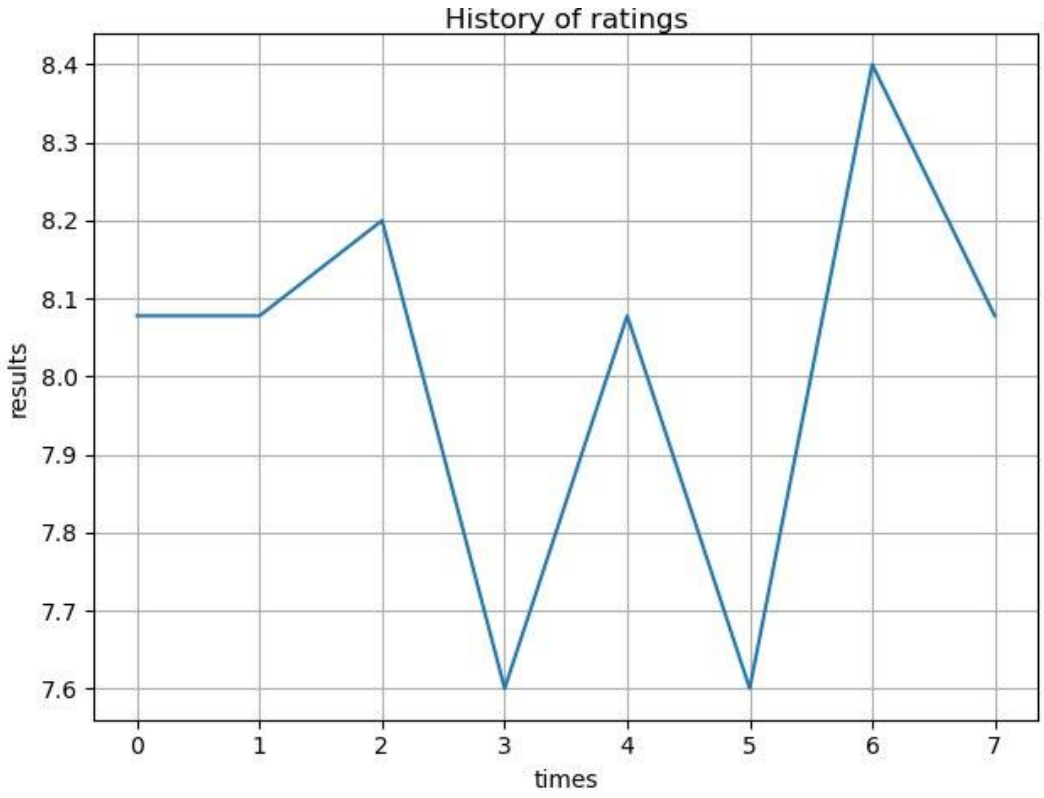


Рис. 8. Таблиця історії зміни цифрової оцінки Інтернет-користувачів

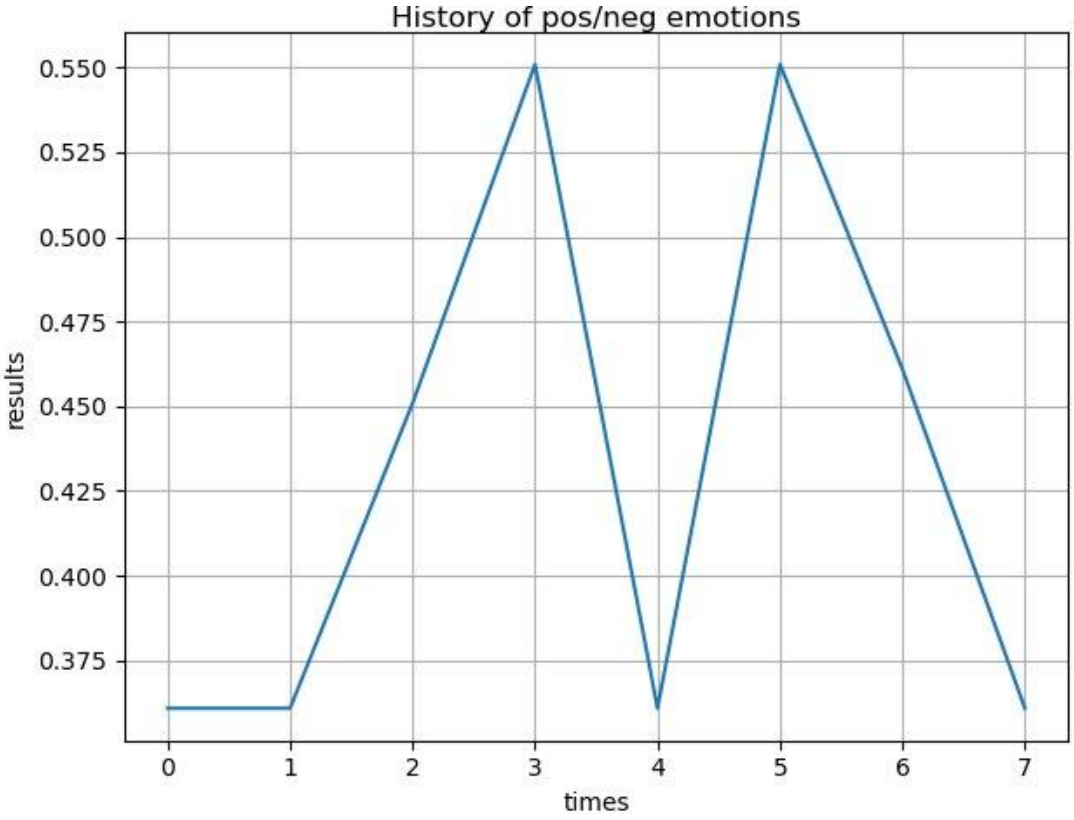


Рис. 9. Таблиця історії зміни оцінки емоцій відгуків за Vader

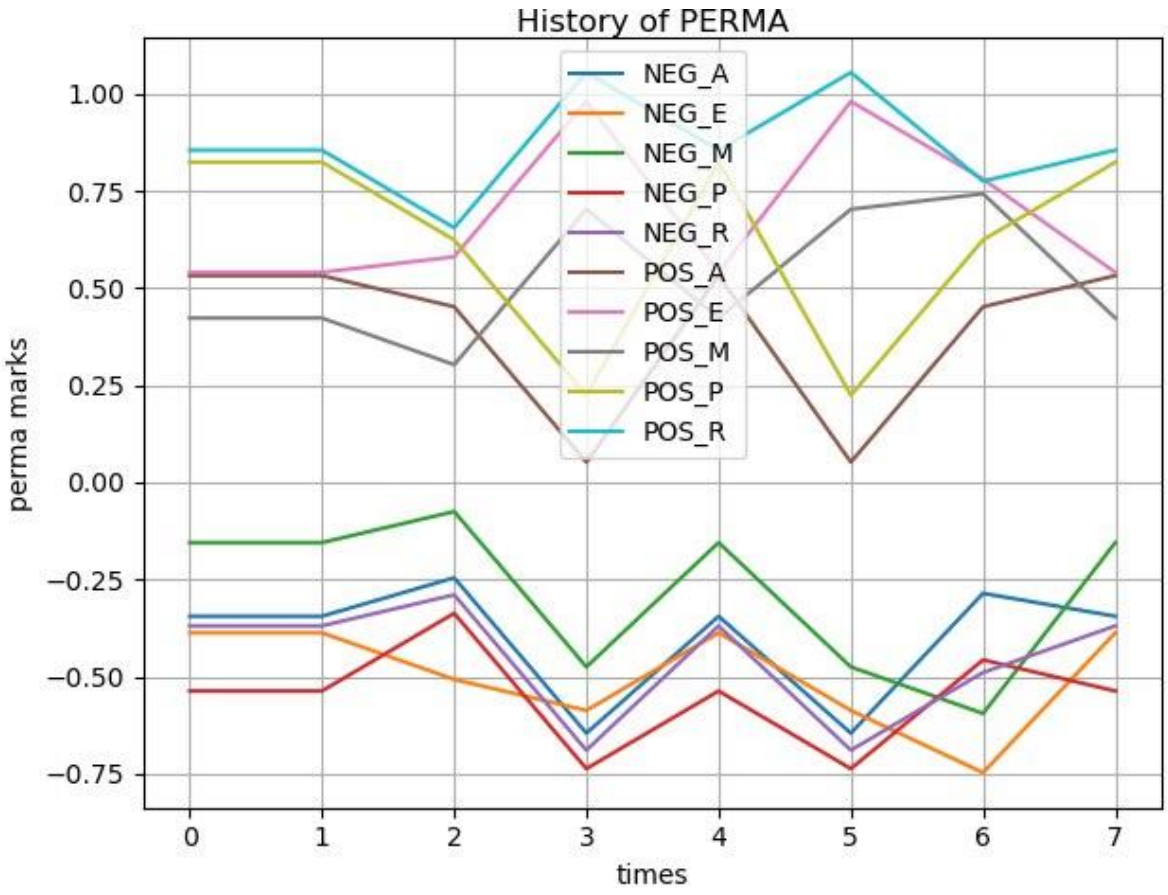


Рис. 10. Таблиця історії зміни оцінки емоцій відгуків за PERMA